Qafqaz University Press

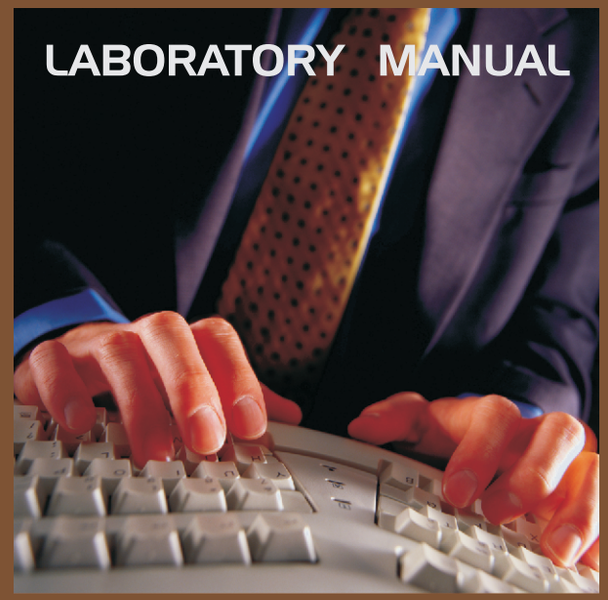
Press No: 42

DIGITAL LOGIC DESIGN

Khalil Ismailov

Khalil Ismailov

# DIGITAL LOGIC DESIGN




Second Edition

Qafqaz University Press
Baku - 2010

Ministry of Education of Azerbaijan Republic

**Institute of Educational Problems**

"Çağ" Educational Corporation

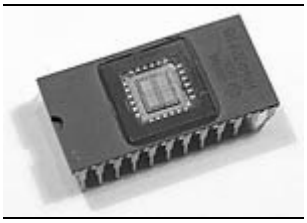
**Qafqaz University**

# DIGITAL LOGIC DESIGN

## LABORATORY MANUAL

**Second Edition**

**Khalil Ismailov**



*Approved by the decision of the Scientific-Methodical Commission's "Informatics and Computer Technologies" Section of the Ministry of Education of Azerbaijan Republic dated January 16, 2007 (minute No. 01)*

Baku, 2010

# DIGITAL LOGIC DESIGN

## LABORATORY MANUAL
**Second Edition**


**Prepared by** **Kh. A. Ismailov, Professor**
*Department of Computer Engineering,*
*Qafqaz University*

**Reviewed by** **T.A. Alizade, PhD**
*The Institute of Cybernetics of the National*
*Academy of Sciences of Azerbaijan Republic*

**S.B. Habibullaev, Associate Professor**
*Department of Computer Technologies and Programming,*
*Azerbaijan State Oil Academy*

**Editor** **A.Z. Melikov, Professor**
Associate Member of the National
Academy of Sciences of Azerbaijan Republic,
The Institute of Cybernetics of the National
Academy of Sciences of Azerbaijan Republic

**Design** **Sahib Kazimov**



*Is published as a Qafqaz University publication by the proposal of the Publishing Committee dated from 04.10.2006 (minute No. Ç-QU-250-300/020) and decision of the Senate dated from 11.10.2006 (minute No. Ç-QU-105-400/23)*

*Book is printed by "Sharg-Qarb" Publishing House.*




***Press No: 42***

# CONTENTS

## PREFACE

*This manual is designed to provide practical laboratory experience for the student of digital electronics. The manual includes experiments on digital fundamentals and design. It is not intended that all the experiments be completed or that every experiment be done in its intirety. Instructors will probably want to make certain exercises optional to the student.*

*Each experiment begins with a set of stated objectives, text references, and required equipment, followed by a procedure for meeting each objective. Most experiments specify logic circuits needed to perform the experiment, while a few require that the student design and draw the circuit(s).*

*This manual contains several appendixes at the end. The students is encouraged to become familiar with the contents of the appendixes early, even though serious reading of much of the information on troubleshooting and digital faults can be deferred until it is needed.*

*Appendix A covers the types of circuit breadboards used in laboratory experiments for temporary circuit testing.*

*Appendix B covers several topics which should be helpful to the student, Included in the material are general hints on proper breadboarding, digital troubleshooting, and information on typical digital system and IC faults.*

*The information about digital ICs and 74 series family cross-reference of 155 series Russian ICs are found in Appendix C.*

*The pin layout for digital ICs used in experiments are found in Appendix D of this manual.*

*The information about seven segment displays of type HDSP is provided in Appendix E.*

## EXPERIMENT GUIDELINES

Here are some guidelines to help you perform the experiments and to submit the reports:

- Read all instructions carefully and carry them all out.
- Ask a demonstrator if you are unsure of anything.
- Record actual results (comment on them if they are unexpected!)
- Write up full and suitable conclusions for each experiment.
- If you have any doubt about the safety of any procedure, contact the demonstrator beforehand.
- **THINK** about what you are doing!

Most experiments involve the use of the power supply, the oscilloscope, a signal generator, the voltage measurement unit and your breadboard.

## GENERATING DIGITAL SIGNALS FOR TESTING

The most common type of signal needed to test logic circuits is a static signal that can be switched HIGH or LOW with a simple single throw (SPST) switch. New students are often confused by how to do this correctly. The problems involved, and the solution, are illustrated in Fig. 0.1.

The problem is illustrated in (a), which shows a logic circuit to be tested, a 5 V power supply and a switch. The logic circuit will be connected to the power supply, and the switch should be used to provide a signal input voltage, $V_{IN}$, to the circuit such that changing the switch position will change the signal input voltage between logic HIGH and LOW. How should the switch be connected? It cannot be connected directly across the supply since this would short the supply when it is closed. An arrangement sometimes tried by new students is shown in (b). In this case, when the switch is open, the input to the circuit is an open circuit so that the input voltage is undefined, which will not produce correct operation of the circuit to be tested.

There are two possible solutions, which are shown in Fig. 0.l (c) and (d). In both of these the switch is in series with a resistor between the terminals of the power supply, but the circuit shown in (c) will not usually be satisfactory and the circuit in (d) is one that should be used. This circuit is referred to in the Laboratories as a *static logic level signal source*. The reasons why the circuit

in (c) will not operate correctly are due to the properties of the inputs to TTL and other integrated circuits. In order for TTL circuits to function correctly the LOW input level must be below 0.8 V while the HIGH input level must be above 2.0 V. The circuit in (c) will always produce a correct HIGH level of 5.0 V when the switch is open and the logic under test draws its current directly from the supply. However, when the switch is closed current will flow *out* of the circuit input through the resistor, R, to ground (R is called a pull-down resistor). This produces a voltage drop across R and a LOW level input voltage, $V_L$, which is above ground. In TTL circuitry current flow when the input voltage is HIGH. Thus an appreciable current flowing through R when the switch is closed can bring the input voltage above the allowable maximum. The only way to avoid this is to make R very small, but this will drain a large current from the power supply when the switch is closed.

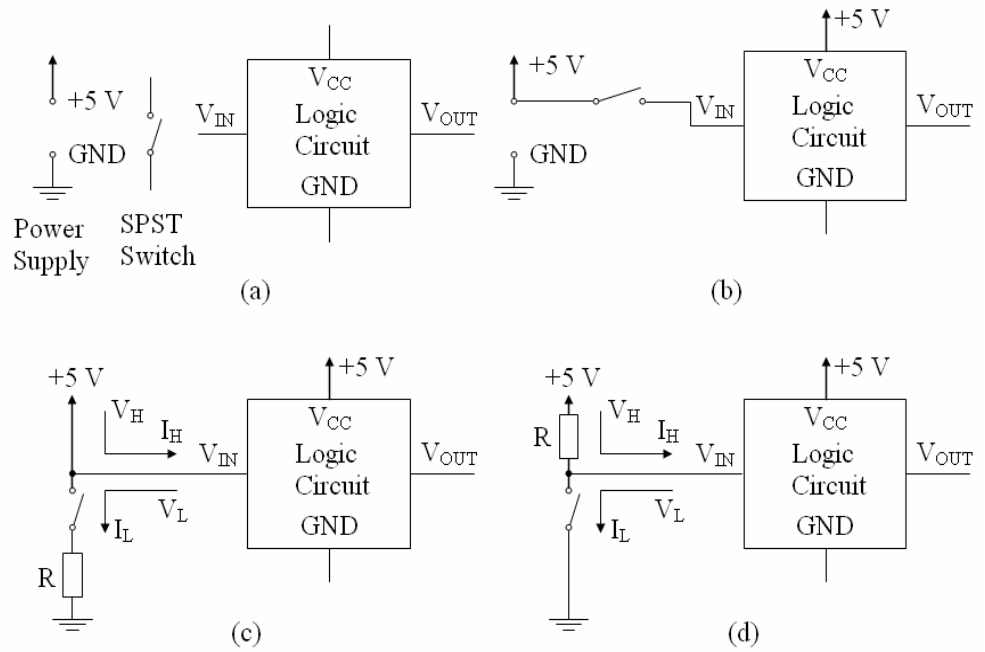



**Fig. 0.1.** Generating a static logic level signal

The circuit shown in (d) functions much better the student still needs to understand its limitations. In this circuit, when the switch is closed, the input to the test circuit is connected directly to ground so that the LOW voltage is at zero volts regardless of the magnitude of the input current. When the switch is

open, the test circuit input voltage is connected to the +5.0 V supply through the resistor, R. Note that whatever input current, $I_H$, is required by the circuit under test must flow through R and will therefore produce a voltage drop reducing the HIGH level input voltage, $V_H$, to a value of $(5 - RI_H)$. There are two factors to consider. First, HIGH level input currents are considerably lower than LOW level input currents due to the nature of the circuitry used in the integrated circuits. Second, the HIGH level input voltage can be allowed to drop to as low as 2.0 V without violating circuit specifications. Both of these mean that, *in most applications,* the resistor, R, can be made reasonably large so that the HIGH level voltage is properly defined without drawing too much current from the power supply when the switch is closed. The resistor, R, is known as *pull-up resistor*. When designing static logic level sources its value must be chosen to produce adequate voltage levels for the circuits to be tested. In almost all cases, values between 1.0 kΩ and 10.0 kΩ will work well. The student is cautioned, however, that if a single switch circuit is to be used to provide multiple inputs to a test circuit, it may be necessary to adjust the value of the pull-up resistor to maintain adequate HIGH level voltages. In building a logic circuit facility the student will need multiple static signal sources. Eight such circuits should be sufficient for most Assignments in the Laboratories.

The second type of signal generating requirement necessary for testing logic circuits is the ability to generate a single pulse from a manual switch. This type of circuit is referred to in the Laboratory equipment lists as a *push button manual pulser*. The pulse produced by the pulser must be "clean" with sharp edges. Also, for greatest versatility, the pulser should be able to produce both HIGH-going as well as LOW-going pulses. A simple SPST switch circuit such as that of Fig. 0.l (d) cannot be used because of the problems *of switch bounce*. When the mechanical contacts of the switch are closed, they actually bounce together a multiple and indeterminate number of times before coming to rest in a closed position. This produces a voltage that switches between HIGH and LOW before setting to a LOW level, so that rather than producing a single pulse, multiple pulses would be produced. In order to avoid this, a single pole double throw (SPDT) switch must be used with some type of a latch circuit. One commonly used circuit is shown in Fig. 0.2.

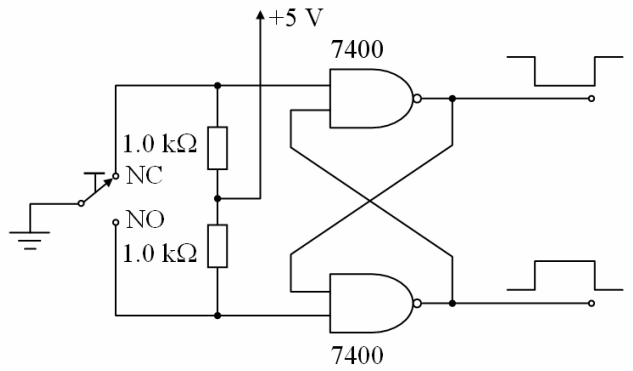



**Fig. 0.2.** Manual pulser circuit

For the push button SPDT switch the center pole is connected to ground. The cross-coupled gates form a NAND latch which has the property that the output level is determined by the last input to become LOW; this removes the switch bounce effects. Note that both LOW-going and HIGH-going pulses are provided at the outputs, with the HIGH-going pulse derived from the gate with its input connected to the normally open (NO) pole of the switch. The 7400 integrated circuit contains four NAND gates. Using this single integrated circuit package with two swatches and four resistors would provide the student with the capability of having two independent pulse sources available for testing circuit designs.

Fig. 0.3 shows a circuit for converting the output of a conventional square wave generator (SWG) so that it is compatible with TTL levels.

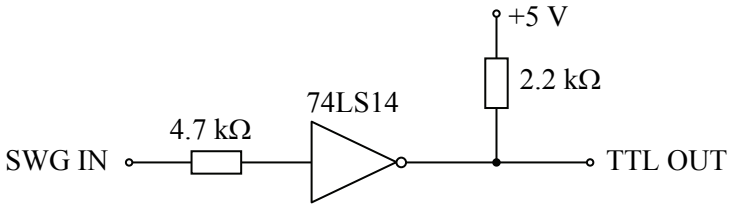



**Fig. 0.3.** Circuit for converting SWG output to TTL levels

## DETECTING LOGIC SIGNALS

In testing their circuit designs students need to be able to determine logic levels at circuit outputs. It is most convenient to have a visual display. Although commercially available logic probes are ideal instruments for this

purpose, the discussion below shows how to use simple light emitting diode (LED) circuits to satisfy most needs.

LEDs are semiconductor diodes that emit light when they are forward biased. The light can be of various colors depending on the semiconductor used in the diode. It is important to remember that LEDs are not made from silicon. When they are forward biased the forward voltage drop is about 1.2 V to 1.4 V. Fig. 0.4 shows three possible ways of using LEDs as logic level indicators.

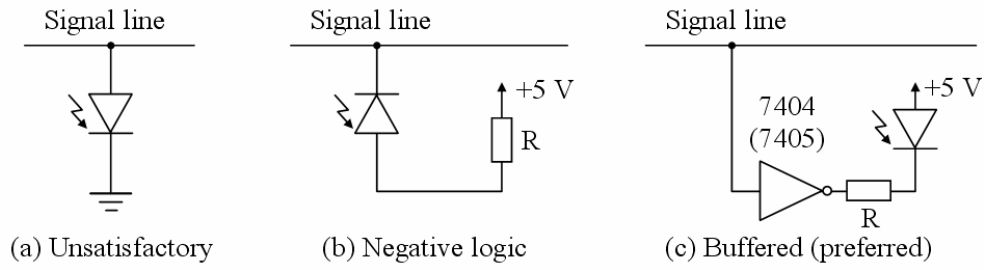



**Fig. 0.4.** LED indicating circuit

Simply connecting the LED between the signal line to be monitored and ground, as shown in (a), is almost always unsatisfactory. The most serious problem is that this connection drops the voltage on the signal line to the diode forward bias voltage of 1.2 V to 1.4 V, which values are between the HIGH and LOW threshold specifications for TTL logic. Thus any other device that is using the signal as input will not function correctly. The circuit shown in (b) connects the diode between the positive supply and the signal line through a current limiting resistor, R. This works well electrically, but it has the disadvantage of giving a negative logic indication of the logic level, i.e. the light is ON when the signal line is LOW. The circuit in (c) is the preferred way of using an LED indicator. In this circuit the LED is buffered from the signal line by a 7404 or 7405 inverter. Thus the LED is ON when the logic level of the signal is HIGH. Again, a current limiting resistor, R, is required. Its value should be chosen to give adequate intensity for the size of LED being used. A value of 330 Ω will give a forward bias current of about 10 mA. This is the circuit that should be used for the Laboratory Assignments when the Equipment and Supplies list calls for LED indicating circuits. Since the 7404 integrated circuit contains six inverters, the student will find it most convenient to use a single 7404 or 7405 package with six resistors and six LEDs to build six indicator circuits which can be kept on the breadboard for general use.

## LOGIC PROBES

A logic probe is a small, hand-held instrument used to indicate the logic level at a point in a digital circuit. It is capable of indicating a logic 0, logic 1, and a level floating between logic 0 and 1. In many cases it is capable of detecting the presence of high-speed pulses. The red clip of the logic probe is to be connected to the power supply (+5 V) of the device under test and the black clip to power supply ground. If probe has a single LED to indicate logic levels the LED should be blinking to indicate that the level at the probe tip is floating. If probe has two LEDs to represent the two logic levels, both should be out. If you have a DMM available, connect its leads between the logic power ground and the probe tip. The DMM should read a value between 1.3 and 1.5 volts. This value is called by various names - "indeterminate", "bad", "invalid", and "floating". All refer to the fact that if a voltage value falls into a range of greater than 0.8 volts and less than 2.0 volts, the value cannot represent a logic 0 or a logic 1. These values all have a tolerance of 20%.

Most logic probes have three LEDs for logic indicators. One lights for a logic 1 input and another for a logic low input. The third lights when the input is pulsing.

## LOGIC PULSERS

A logic pulser generates digital pulses. Like the logic probe, the pulser uses the logic power supply to get its own power. The tip of the pulser is placed on a circuit node where an injected pulse is desired. The pulser senses the logic state of the node and generates a pulse that will attempt to drive the node to the opposite state. For example, if the logic pulser is placed into the circuit where there is a logic low signal level, it will automatically generate a high level signal. It is a valuable aid in troubleshooting, since it permits the triggering of gates and other devices without removing them from their circuits.

## POWER SUPPLIES

Whenever possible a fixed 5 V power supply should be used. This is not only more convenient but it avoids the possibility of circuit damage due to incorrect voltage level settings. However, there are some assignments that require other voltage levels (+15 V, -15 V) for the circuit under test, and the student may find it necessary to use a dual variable supply for these. These should be used and

adjusted carefully because TTL digital circuits are easily damaged if the power supply voltage varies significantly from the specified 5 V level.

TTL circuits require an accurate, well-regulated voltage supply of 5 V + 0.25 V.

## DIGITAL LOGIC TRAINER Y-0039

Digital Logic Trainer Y-0039 (Fig. 05) provides all the basic functions necessary for learning digital logic fundamentals and prototyping digital logic circuits.

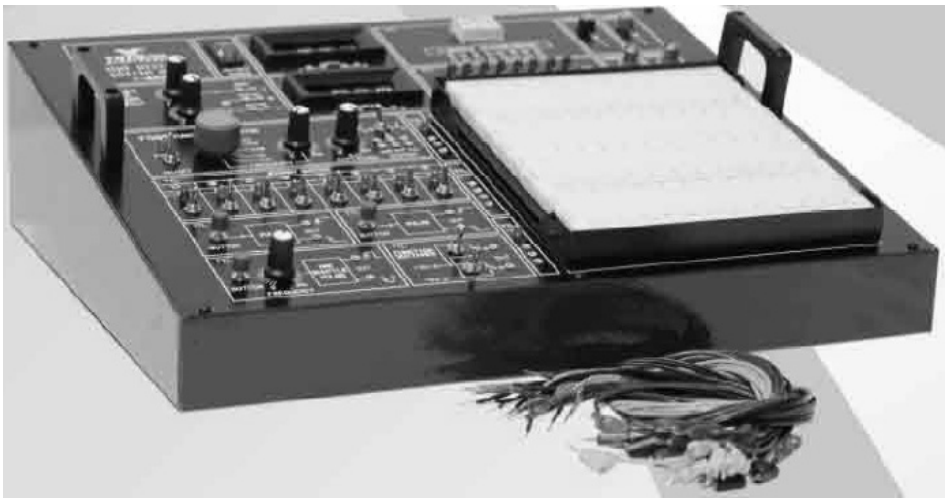


**Fig. 05.** Digital Logic Trainer Y-0039

### Function of Controls, Connectors, and Indicators

Before turning this trainer on, familiarize yourself with the controls, connectors, indicators, and other features described below. The following descriptions are keyed to the items called out in Fig. 0.6.

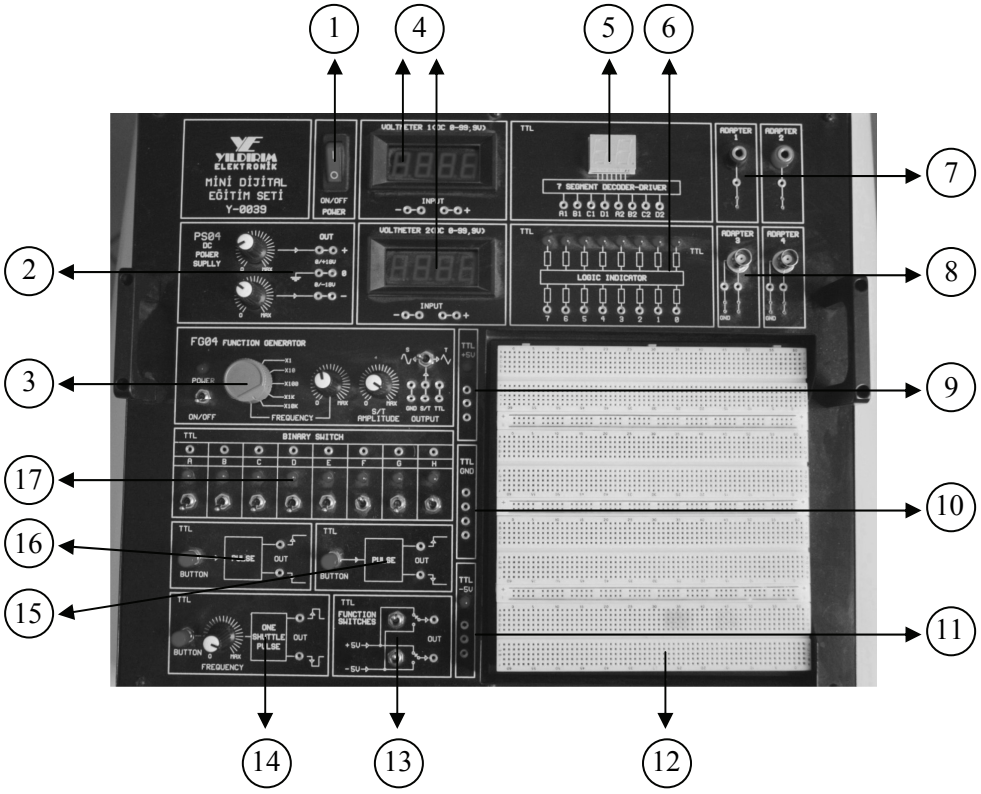



**Fig. 0.6** The front panel of Y-0039

1 – Power ON-OFF Switch
2 – 0 – (+18 V), 0 – (–18 V) Adjustable DC Power Source (PS04)
3 –1 Hz 100 Hz Function Generator (FG04)
4 – Voltmeter
5 – 2x7-Segment Decoder
6 – 8-Bit TTL Logic Indicator
7 – Adapter 1
8 – Adapter 2
9 – (+5 V) TTL Power Source
10 – TTL Ground (GND)
11 – (–5 V) TTL Power Source
12 – Breadboard
13 – TTL Function Switches
14 – TTL One-Shuttle Pulse Circuit (One-Shot Multivibrator)
15 – TTL Single-Edge Pulse Generator
16 – TTL Single-Edge Pulse Generator
17 – Binary Switches

## Specifications

| | |
|---|---|
| Fixed DC Power Supply | Voltage range (TTL): +5V and –5V.<br>Maximum current output: 1 A |
| Variable DC Power Supply | Voltage range: +0 V ~ +18 V, 0 V ~ –18 V<br>Maximum current output: 0.5 A |
| Voltmeter | Two 3½ digit LED displayed voltmeters with range from 0 to 99.9 V |
| Variable Clock Generator | Four frequency ranges:<br>1 Hz to 110 Hz<br>10 Hz to 1100 Hz<br>100 Hz to 11 kHz<br>950 Hz to 100 kHz<br><br>Output – sinusoidal, triangle and square waves. Sinusoidal and triangle output range are adjustable from 0.1 Vpp to +10 Vpp. TTL output level is a fixed 5 V. |
| Logic Indicators | 8 sets of independent LEDs, indicating high and low logic states |
| Data Switches | One 8-bit DIP switches giving 8-bit TTL level output |
| TTL Function Switches | Two switches, each having debounced +5 V and -5 V outputs (Up position +5 V, down position -5 V) |
| One-Shuttle Pulse Circuit | Produces negative and positive pulses for TTL applications |
| Adapter Output | Allows to connect DMMs, oscilloscopes and function generators to trainer elements and circuits |
| Seven-Segment Displays | Two sets of independent 7-segment displays with BCD, 7-segment decoder/driver and decimal point input terminal, input with 8-4-2-1 code |
| Removable Solderless Breadboard | 2800 holes, accepting all DIP devices, components with leads and solid wires (0.3 mm to 0.8 mm) |
| Accessories | Power lead, connecting leads, fuse, dust cover, and user manual |
| Power Supply | 220V AC ±10%, 50 Hz |

EXPERIMENT

# 1

# DIGITAL LOGIC CIRCUITS

## OBJECTIVES

1. To determine by experiment the function table for the digital logic gates.
2. To construct a simple combinational logic circuit and prove its function table.

## EQUIPMENT REQUIRED

Digital Logic Trainer
Digital Multimeter (DMM)
Dual Trace Oscilloscope
1 each, 7400, 7402, 7404, 7408, and 7432 integrated circuits

## BASIC INFORMATION

In digital circuitry a *gate* is a circuit with two or more inputs and a single output. If the inverter is considered to be a gate, it is an exception to the rule with a single input. Not all people consider the inverter to be a gate. There are two basic gate circuits: OR and AND (Fig. 1.1).


**Fig. 1.1.** OR and AND gates


**Fig. 1.2.** NOR and NAND gates

When the inverter is added to the AND and OR gates, as in Fig. 1.2, NOT AND (NAND) or NOT OR (NOR) circuits result, making two more types of gates.

Combinations of basic gates can be used to perform complex logic operations in computers and other digital equipment. Fig. 1.3 is an example of combining AND and OR gates. To analyze this circuit, it would be necessary to consider what happens for all possible inputs.

It would be rear today to find a logic gate built from discrete transistors. Integrated circuits (ICs) have been used exclusively for logic gates for the past several years. The most widely used digital circuit family is the transistor-transistor logic (TTL) family. Examples are a 5400 or a 7400 series. The 5400 series is a military version of the commercial 7400 series. The AND gate is a 7408. The OR gate is a 7432. The 5 V $V_{CC}$ attaches to pin 14 in both devices. Likewise, the ground pin is pin 7. This is not always true, so refer to data manuals or published pin-outs before you make any connections. Also note that, as in all other ICs, the location of pin 1 is indicated by either a dot over the pin or the notch on one end of the package.
Note that in TTL, a disconnected input is equivalent to the input set at "high".

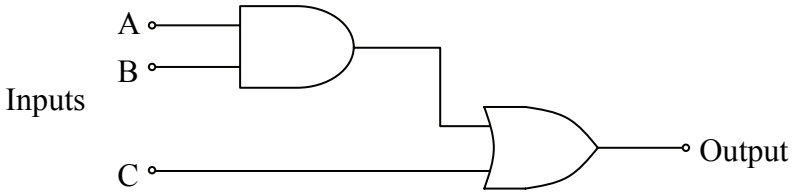



**Fig. 1.3.** Combinational logic circuit

## PROCEDURE

### OR Gate

1. Connect the 7432 OR gate as shown in Fig. 1.4. Note that +5 V and ground connections are not shown on logic diagrams, but you must be sure to connect them.

   For the reminder of this experiment you may use wire connections to the switches on the Digital Logic Trainer Y-0039 to connect the inputs to "Logic 1" or ground for the 1 and 0 inputs.

2. Connect the output to the DMM.

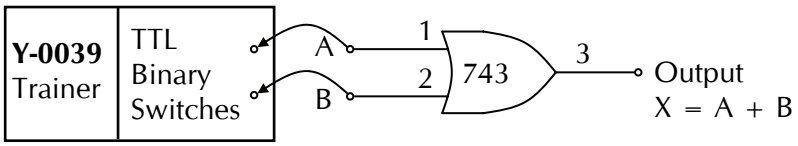



**Fig. 1.4**. Experimental OR gate circuit

3. You will now verify the OR operation by setting inputs A and B to each set of logic values listed in the Table 1.1. Record the output voltage observed and convert the output voltage to a logic level. Use

   $$0\text{ V} - 0.8\text{ V} = 0 \text{ and } 2\text{ V} - 5\text{ V} = 1$$

   for the conversions and record your observations in Table 1.1.
4. Disconnect the DMM from the circuit, and use a TTL Logic Indicator (LED Monitor) to observe the output. Repeat step 3 using the conversion rule LED OFF (unlighted) = 0 and LED ON (lighted) = 1. Record your observations in Table 1.1.
5. Disconnect one of the inputs, and set the remaining one to 0. Is the output level 0 or 1? Based on your observation and knowledge of the OR operation, what level does the unconnected input act like? ___ .

**Table 1.1.** OR gate circuit experimental results

| Data Switches | | DMM (V) | Output Logic Level (0/1) |
|---|---|---|---|
| A | B | | |
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

## AND Gate

1. Connect the circuit in Fig. 1.5. Do not forget the power supply connections.

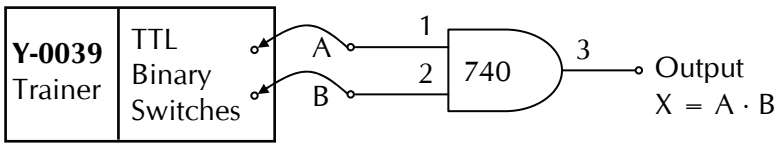



**Fig. 1.5**. Experimental AND gate circuit

2. Connect the output to the TTL Logic Indicator.

   You will now verify the AND operation by setting inputs A and B to each set of logic values listed in the Table 1.2. Record the output level observed on the TTL Logic Indicator using Table 1.2.

**Table 1.2.** AND gate circuit experimental results

| Data Switches | | Output of TTL Logic Indicator (ON/OFF) | Output Logic Level (0/1) |
|---|---|---|---|
| A | B | | |
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

3. Disconnect one of the inputs, and set the remaining one to 1. Note the logic level indicated by the TTL Logic Indicator. Based on your observation, what logic level does the unconnected input act like? ___ .

**NOT Gate**

1. Connect the circuit in Fig. 1.6. Do not forget the power supply connections.
2. Connect the output to the TTL Logic Indicator.

   You will now verify the NOT operation by setting input A to each set of logic values listed in the Table 1.3. Record the output level observed on the TTL Logic Indicator using Table 1.3.

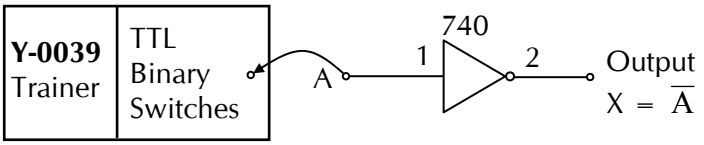



**Fig. 1.6**. Experimental inverter circuit

**Table 1.3.** NOT gate circuit experimental results

| Data Switch (A) | Output of TTL Logic Indicator (ON/OFF) | Output Logic Level (0/1) |
|---|---|---|
| 0 | | |
| 1 | | |

## NOR Gate

1. Connect the circuit in Fig. 1.7. Do not forget the power supply connections.

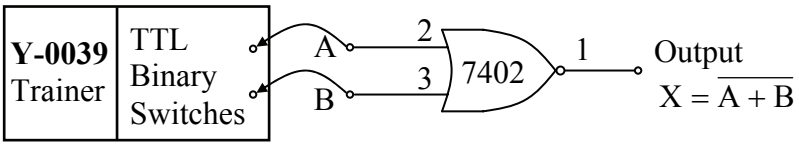



**Fig. 1.7**. Experimental NOR circuit

2. Connect the output to the TTL Logic Indicator. Set the toggle switches to each input combination listed in the Table 1.4, observe and record the output state of the TTL Logic Indicator in the Table 1.4.

**Table 1.4.** NOR gate circuit experimental results

| Data Switches | | Output LED Monitor (ON/OFF) | Output Logic Level (0/1) |
|---|---|---|---|
| A | B | | |
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

Verify that your results agree with the truth table for NOR gate.

3. Disconnect input B from the toggle switch. Set toggle switch A alternately to 0 and 1, and observe the effect on the output. Based on your observation, the disconnected NOR input acts like a ___ input level.

4. Connect the output (GND and TTL) of a FG04 Function Generator on the Y-0039 Digital Logic Trainer to input B and set the generator to 1 kHz (×1k position). Disconnect the TTL Logic Indicator from the NOR output, and connect one of the vertical inputs of the oscilloscope in its place. Connect the other vertical input to the output of the FG04 Function Generator, and trigger on this channel. Set input A alternately to 0 and 1, and observe the effect on the output. Sketch the waveform displayed on the oscilloscope for both settings of switch A using Timing Diagrams 1.1 and 1.2.


**Timing Diagram 1.1**


**Timing Diagram 1.2**

### NAND Gate

1. Connect the circuit in Fig. 1.8. Do not forget the power supply connections.

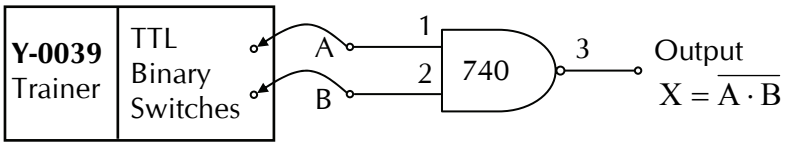



**Fig. 1.8**. Experimental NAND circuit

2. Connect the output to the TTL Logic Indicator.
   Set the toggle switches to each input combination listed in the Table 1.5, and record your observations of the output monitor in the Table 1.5.

**Table 1.5**. NAND gate circuit experimental results

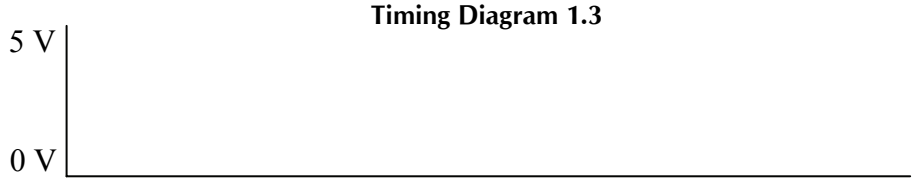
| Data Switches | | Output of the TTL Logic Indicator (ON/OFF) | Output Logic Level (0/1) |
|---|---|---|---|
| A | B | | |
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

Verify that your results agree with the truth table for NAND gate.

3. Disconnect the toggle switch from input B of the NAND gate. What will the state of output X be when A = 0? ____ . When A = 1? ____ . Verify your results.
4. Connect the output (GND and TTL) of a FG04 Function Generator on the Y-0039 Digital Logic Trainer to input B and set the generator to 1 kHz (×1k position)

   Remove the output connection to the TTL Logic Indicator, and connect one of the vertical inputs of the oscilloscope in its place. Connect the output of the function generator to the other vertical input of the oscilloscope, and trigger internally using this channel.
5. Set input A alternately to 0 and 1, and observe the effect on the output. Draw the waveforms displayed on the oscilloscope for each setting of a using Timing Diagrams 1.3 and 1.4.


**Timing Diagram 1.3**




**Timing Diagram 1.4**

## Combinational Logic

1. Wire the circuit shown in Fig. 1.9. Do not forget the power supply connections.

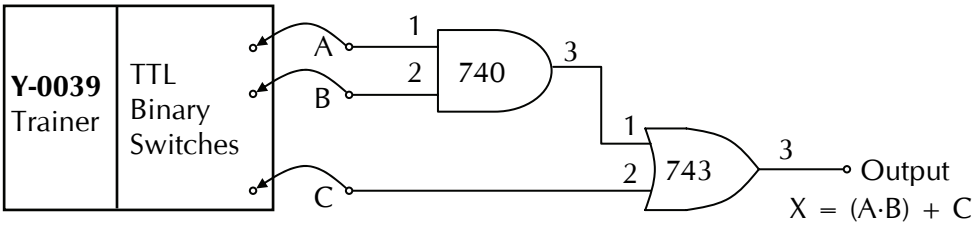



**Fig. 1.9.** Experimental combinational logic circuit

2. Connect the TTL Logic Indicator to output.
3. Set the toggle switches to each input combination in Table 1.6 and observe the output.

**Table 1.6.** Combinational circuit experimental results

| Data Switches | | | Output LED Monitor (ON/OFF) | Output Logic Level (0/1) |
|---|---|---|---|---|
| A | B | C | | |
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

EXPERIMENT

# 2

# SIMPLIFICATION OF BOOLEAN FUNCTIONS

## OBJECTIVE

To use Boolean theorems to simplify logic circuits.

## EQUIPMENT REQUIRED

Digital Logic Trainer
1 each, 7400, 7402, 7404, 7408, 7410, 7411, 7427 and 7432 integrated circuits

## BASIC INFORMATION

One of the more common tasks of digital design is circuit simplification. Although formal techniques exist for the methodical simplification of Boolean circuits and corresponding expressions, the designer should be able to use the fundamental laws and basic relationships of Boolean algebra to analyze a given circuit and find a simplified expression for its output which will lead to a simpler circuit realization. These skills are developed by knowing the necessary laws and relationships, and gaining experience in their application.

The fundamental laws and basic relationships are summarized below.

## FUNDAMENTAL LAWS:

(1) COMMUTATION:
- **(a) $A + B = B + A$**
- **(b) $A \cdot B = B \cdot A$**

(2) ASSOCIATION:
- **(a) $A + (B + C) = (A + B) + C$**
- **(b) $A(BC) = (AB)C$**

(3) DISTRIBUTION:
- **(a) $A + BC = (A + B)(A + C)$**
- **(b) $A(B + C) = AB + AC$**

**BASIC RELATIONSHIPS:**

(4) OPERATIONS WITH 1 AND 0:

**(a) $A \cdot 0 = 0$**
**(b) $A \cdot 1 = A$**
**(c) $A + 0 = A$**
**(d) $A + 1 = 1$**

(5) OPERATIONS WITH A SINGLE VARIABLE:

**(a) $A \cdot A = A$**
**(b) $A + A = A$**
**(c) $A \cdot \bar{A} = 0$**
**(d) $A + \bar{A} = 1$**
**(e)** $\overline{\overline{A}} = A$

(6) MULTIVARIABLE OPERATIONS:

(a) $A + AB = A$
**(b) $A + \bar{A}B = A + B$**
**(c) $\bar{A} + AB = \bar{A} + B$**
**(d) $(A + B)(C + D) = AC + AD + BC + BD$**

(7) DEMORGAN'S THEOREMS:

**(a)** $\overline{A + B} = \overline{A} \cdot \overline{B}$
**(b)** $\overline{A \cdot B} = \overline{A} + \overline{B}$

When applying these equations to the simplification of a given logic circuit it is best to work your way from the inputs to the outputs developing simplified expressions for the intermediate points in the circuit. This is especially important as circuits get more complicated. If simplification is not done as the circuit is analyzed the expressions for the outputs are often so complicated that it is difficult to see the simplifying combinations.

Consider the circuit shown in Fig. 2.1 as an example. Note that intermediate points in the circuit have been labelled as p, q, r, s, t. Logic expressions will be derived and simplified at these points as we work our way from inputs to the output, X. The process is outlined below.

$p = A + B$
$q = \bar{A}(A + B) = \bar{A}A + \bar{A}B = \bar{A}B$
$r = AB$
$s = BC$
$t = rs = (AB)(BC) = ABBC = ABC$
$X = q + t = \bar{A}B + ABC = B(\bar{A} + AC) = B(\bar{A} + C) = \bar{A}B + BC$

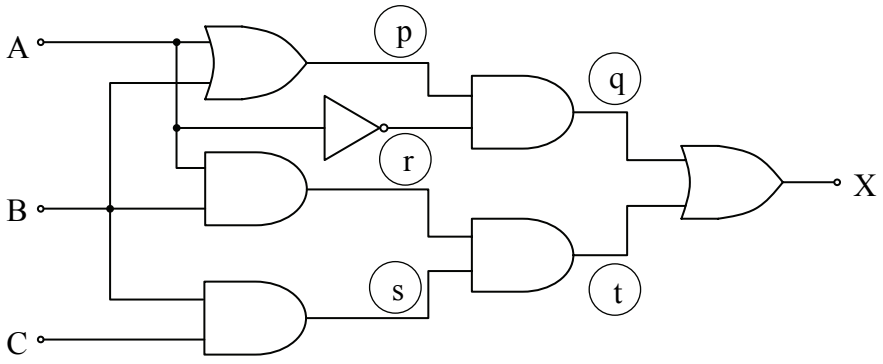



**Fig. 2.1.** Example circuit simplification

A simplified circuit can now be drawn as shown in Fig. 2.2.

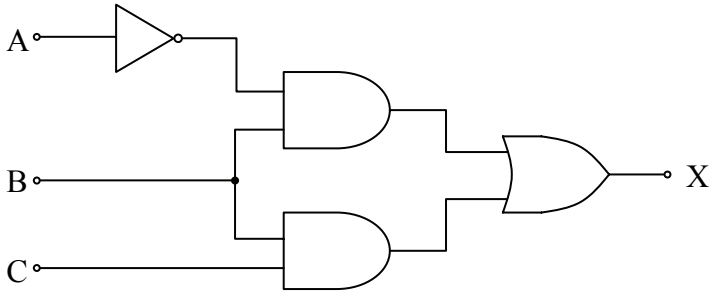



**Fig. 2.2**. Simplified circuit

Note that the circuit of Fig. 2.2 has 3 fewer gates than the original circuit. An even simpler circuit could be realized if the parentheses had not been removed in the last step of the analysis. In this case the output is represented as:

$$X = B(\bar{A} + C)$$

The corresponding circuit is shown in Fig. 2.3 below.

The logical properties of all three circuits above are identical, which could be demonstrated by comparing the truth tables of the outputs. However, their electronic properties may be different.

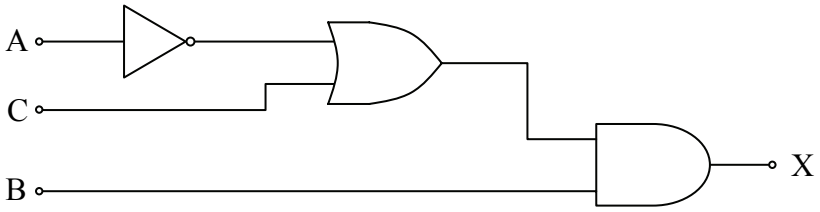



**Fig. 2.3**. Further simplified circuit

## PROCEDURE

### Assignment 2.1

(a) Examine the logic circuit in Fig. 2.4, and write the Boolean expression for output X: _____ .

(b) Make a truth table for expression X using Table 2.1.

(c) Construct the circuit of Fig. 2.4 on the circuit board. Connect toggle switches (TTL Binary Switches on Y-0039 Digital Logic Trainer) to inputs A, B, and C. Connect X to a TTL Logic Indicator on Y-0039 Digital Logic Trainer.

(d) Verify the operation of your circuit by setting the toggle switches to each set of input values in Table 2.1 and comparing the outputs observed to the corresponding outputs in the table.

(e) In the space provided below, simplify X using Boolean theorems. List the theorem used in each step of the simplification.

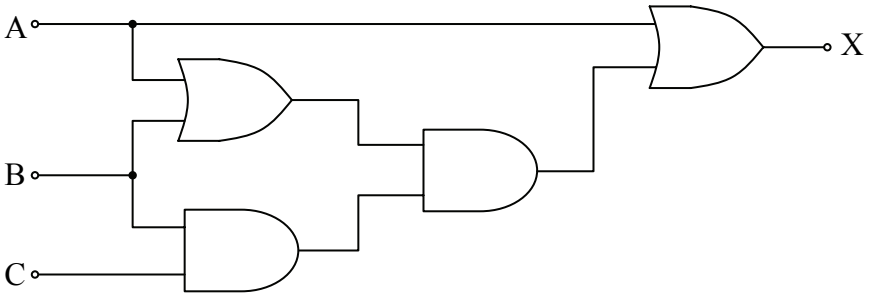



**Fig. 2.4.** Circuit for Assignment 2.1

**Table 2.1**. Output logic level at X for an original and simplified circuits (three inputs)

| Inputs | | | Output Logic Level at X | |
|---|---|---|---|---|
| A | B | C | For an Original Circuit | For a Simplified Circuit |
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

(f) Draw the logic diagram for the simplified expression.

**Assignment 2.2**

(a) Examine the logic circuit in Fig. 2.5, and write the Boolean expression for output X: _____.

(b) Make a truth table for expression X using Table 2.2.

(c) Construct the circuit of Fig. 2.5 on the circuit board. Connect toggle switches to inputs A, B, C, and D. Connect X to a TTL Logic Indicator.

(d) Verify the operation of your circuit by setting the toggle switches to each set of input values in Table 2.2 and comparing the outputs observed to the corresponding outputs in the table.

(e) In the space provided below, simplify X using Boolean theorems. List the theorem used in each step of the simplification.

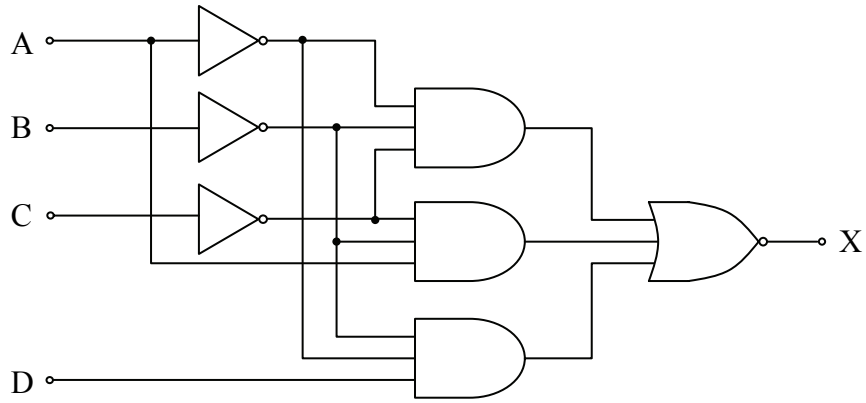



**Fig. 2.5**. Circuit for Assignment 2.2

**Table 2.2.** Output logic level at X for an original and simplified circuits (four inputs)

| Inputs | | | | Output Logic Level at X | |
|---|---|---|---|---|---|
| A | B | C | D | For an Original Circuit | For a Simplified Circuit |
| 0 | 0 | 0 | 0 | | |
| 0 | 0 | 0 | 1 | | |
| 0 | 0 | 1 | 0 | | |
| 0 | 0 | 1 | 1 | | |
| 0 | 1 | 0 | 0 | | |
| 0 | 1 | 0 | 1 | | |
| 0 | 1 | 1 | 0 | | |
| 0 | 1 | 1 | 1 | | |
| 1 | 0 | 0 | 0 | | |
| 1 | 0 | 0 | 1 | | |
| 1 | 0 | 1 | 0 | | |
| 1 | 0 | 1 | 1 | | |
| 1 | 1 | 0 | 0 | | |
| 1 | 1 | 0 | 1 | | |
| 1 | 1 | 1 | 0 | | |
| 1 | 1 | 1 | 1 | | |

(f) Draw the logic diagram for the simplified expression.

**Assignment 2.3**

(a) Simplification using DeMorgan's theorems: Draw a logic diagram for the expression:

$$F = \overline{\left(\overline{A} + \overline{B}C\right)} \cdot \overline{\left(\overline{A} + \overline{B}\,\overline{C}\right)}$$

(b) Construct the circuit using the diagram you drew in step (a). Connect toggle switches to inputs A, B, and C and a TTL Logic Indicator to the circuit output. Set the toggle switches to each input combination listed in Table 2.3 and record the output value observed.

**Table 2.3**. Output logic level at F for an original and simplified circuits (three inputs)

| Inputs | | | Output Logic Level at F | |
|---|---|---|---|---|
| A | B | C | For an Original Circuit | For a Simplified Circuit |
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

(c) Simplify F using the DeMorgan's Theorems.

(d) Draw the logic diagram for the simplified expression.

(e) Construct the simplified circuit, and record the output values in the appropriate column of Table 2.3. Verify the results in both columns for F.

EXPERIMENT

# 3

# BASIC DESIGN OF COMBINATIONAL CIRCUITS

## OBJECTIVES

1. Design combinational circuits using a verbal description of the requirements.
2. Design combinational circuits using truth tables to specify requirements.

## EQUIPMENT REQUIRED

Digital Logic Trainer
Digital IC chips, as required

## BASIC INFORMATION

Many simpler combinational circuits can be designed directly from a verbal description of the requirements. For example, consider the following problem. A circuit must be designed to start a pump motor. The motor will start if a HIGH logic level is applied to a control relay. This motor runs in an environment which is potentially hazardous due to the possibility of explosive vapors collecting in the confined space in which the motor is located. For this reason, a fume sensor has been installed in this space which produces a HIGH level output if hazardous fumes are present. The motor controls the pumping of fluid into a storage tank and it should operate automatically whenever the level in the tank reaches a minimum value. There is a level detector in the tank which produces a LOW output when the level reaches the minimum value. It must also be possible for an operator to start the motor manually by activating a switch which produces a HIGH output. However, the motor should not be started under any conditions when hazardous fumes are present. All signal levels are TTL compatible.

How can the motor control circuit be designed? If the situation above can be described by an exact and concise statement, it will be possible to develop a corresponding Boolean equation. When is it required to start the pump? The

pump must be started when the manual switch is activated AND there are NO fumes OR when the level detector is activated AND there are NO fumes. Note that a corresponding Boolean equation could be written by assigning symbols to the signals involved and taking into account the assertion levels of the signals. Assume the following:

P = circuit output to motor relay, HIGH for on.
M = output from manual switch, HIGH when activated.
L = output from level detector, LOW when at minimum.
F = output from fume sensor, HIGH when fumes present.

With these definitions, the requirements statement above can be translated directly into the following equation.

$$P = M\,\overline{F} + \overline{L}\,\overline{F} \qquad (3.1)$$

A circuit could now be designed to realize this function. It should be noted in the above equation that the L variable must be complemented since it is active low.

One of the greatest difficulties in trying to use a verbal description as the basis for design is insuring that the description is complete and accurate. This is especially true as design requirements become more complex. In many cases the best approach is to make an exhaustive list of the required output values for all possible combinations of the inputs, i.e., to make a truth table for the required outputs. Once the truth table is constructed, it is easy to write a Boolean equation for the required outputs and design the circuits accordingly.

Table 3.1 shows the truth table constructed for the motor control problem described above. This is easily obtained by analyzing each of the eight possible combinations of the input variables to determine the required value of the output.

**Table 3.1**. Motor control circuit truth table

| M | L | F | P |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Once the truth table has been determined for a function, the Boolean equation can be written by forming a product term for each input combination for which the value of the function is one, and then logically adding the terms together. Note that each variable is present in the product terms, in its uncomplemented form if the value of the variable is zero. This produces a functional expression in the standard sum of products form (SOP). Often this is not the simplest possible expression. Equation (3.2) below shows the expression derived from the motor control circuit truth table. Note that this is a more complex expression than that derived from the verbal description.

$$P = \overline{M}\ \overline{L}\ \overline{F} + M\ L\ \overline{F} \qquad (3.2)$$

When designing from truth tables it is sometimes found that the function has a large number of 1's as shown in Table 3.2 for a function, X, of three input variables (A, B, C). If the procedure used above were applied in this case, the SOP expression would contain many terms. A better approach is to generate an expression for the complement of the function; this expression can then be complemented to obtain the required function. This is shown in equation (3.3).

$$\overline{X} = \overline{A}BC$$

$$X = \overline{\overline{A}BC} = A + \overline{B} + \overline{C} \qquad (3.3)$$

**Table 3.2**. Using function complement

| A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

## PROCEDURE

For one of the assignment given by the instructor:

1. Write the Boolean equation
2. Simplify the Boolean function
3. Draw the logic diagram of the circuit using required logic gates with a minimum number of ICs.
4. Obtain the necessary chips, construct the circuit and test it for proper operation by verifying the conditions stated above.

## LABORATORY ASSIGNMENTS

### Assignment 3.1

Design a circuit to activate an alarm in an industrial process control application. The alarm, which is activated by a HIGH level signal, is to depend on three variables, pressure (P), temperature (T), and level (L), which are monitored in the process. Assume that setpoint values have been assigned for each variable so that the Boolean variables are either 1 or 0 as the physical quantities are above or below the setpoint values. The alarm conditions are:

1. LOW level with HIGH pressure
2. HIGH level with HIGH temperature
3. HIGH level with LOW temperature and HIGH pressure

Design the corresponding alarm control circuit using NAND gates and inverters.

### Assignment 3.2

The design of the control circuitry for the CPU (central processing unit) of a computer requires that a control signal be generated which depends on a clock signal, CLK, and two state signals, T1 and T2. The control signal must go LOW only when CLK is HIGH and T1 is LOW, or when CLK is LOW and T2 is HIGH.

Design the circuit to generate the control signal. Use only NOR gates and inverters.

### Assignment 3.3

Design a circuit to realize the function X in Table 3.3. Use only NAND gates and inverters.

**Assignment 3.4**

Repeat Assignment 3.3 for the function Y in Table 3.4. Use only NOR gates and inverters.

**Table 3.3**

| A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**Table 3.4**

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**Assignment 3.5**

Design a majority logic which is a digital circuit whose output is equal to 1 if the majority of the inputs are 1's. The output is 0 otherwise. Design and test a 3-input majority circuit using NAND gates with a minimum number of IC's.

**Assignment 3.6**

An analog-to-digital converter is monitoring the DC voltage of a 12 V storage battery on an orbiting spaceship. The converter's output is a four-bit binary number, ABCD, corresponding to the battery voltage in steps of 0.75 V, with A as the MSB. The converters binary outputs are fed to a logic circuit (Fig. 3.1) that is to produce a HIGH output as long as the binary value is greater than $1000_2 = 8_{10}$, that is, the battery voltage is greater than $8 \times 0.75$ V = 6 V.

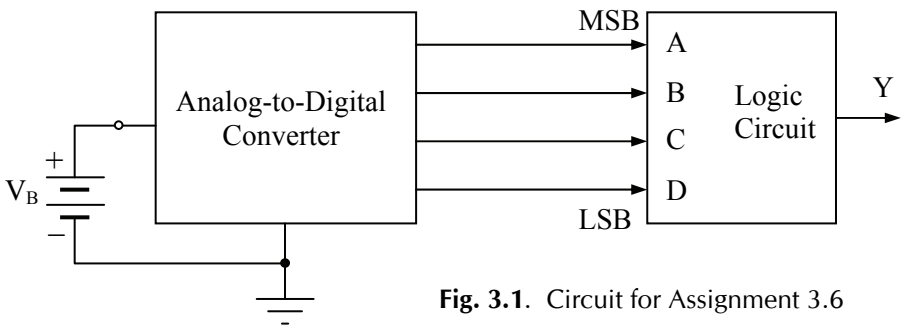



**Fig. 3.1**. Circuit for Assignment 3.6

In the space provided below, write and simplify the Boolean expression, draw corresponding circuit and define output function (assignments 3.5 and 3.6) for the given assignment.

**Assignment No ___ .**

**Boolean expression:**

**Designed logic circuit:**

**Truth tables:**

**For Assignment 3.5**

| A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

**For Assignment 3.6**

| A | B | C | D | Y |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | |
| 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 0 | |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 1 | |
| 1 | 0 | 1 | 0 | |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 0 | 1 | |
| 1 | 1 | 1 | 0 | |
| 1 | 1 | 1 | 1 | |

EXPERIMENT

# 4

# KARNAUGH MAPS AND DESIGN MINIMIZATION

## OBJECTIVES

1. Define circuit level and relate it to circuit speed.
2. Construct the Karnaugh map of any given function of 2, 3 or 4 variables.
3. Use Karnaugh map simplification to design minimized 2-level combinational circuits.

## EQUIPMENT REQUIRED

Digital Logic Trainer
Digital IC chips, as required

## BASIC INFORMATION

In addition to providing problem solutions that are functionally correct, the circuit design process must also provide solutions that satisfy other constraints. In logic circuit design these constraints are typically minimized cost, minimized power dissipation and maximized speed. Minimized cost and power dissipation typically imply minimizing the chip count for any given logic family. In combinational circuits, maximizing the circuit speed implies minimizing the circuit "logic level", again for any given logic family.

The logic level of a circuit is measured by the maximum number of gates between inputs and outputs. The greater the logic level the slower the circuit because each gate causes a timing delay, called the propagation delay, before the output can respond to changes in the inputs. Fig. 4.1 (a) shows a 4-level circuit which realizes the function, X = A(B + CD) + E. The circuit shown in Fig. 4.1 (b) is a 2-level circuit which realizes the same function in the form X = AB + ACD + E. This circuit would be faster than the circuit in (a) because there are fewer propagation delays between inputs and output. It should be noted that the 2-level circuit corresponds to a sum-of-products (SOP) representation of the function. A design must often effect a compromise

between chip count and speed depending on the design environment. For example, if the function, X, were being generated in a system which had unused 2-input gates available, but no 3-input gates, the 4-level circuit of (a) may be preferred since it could be built without requiring any additional chips.

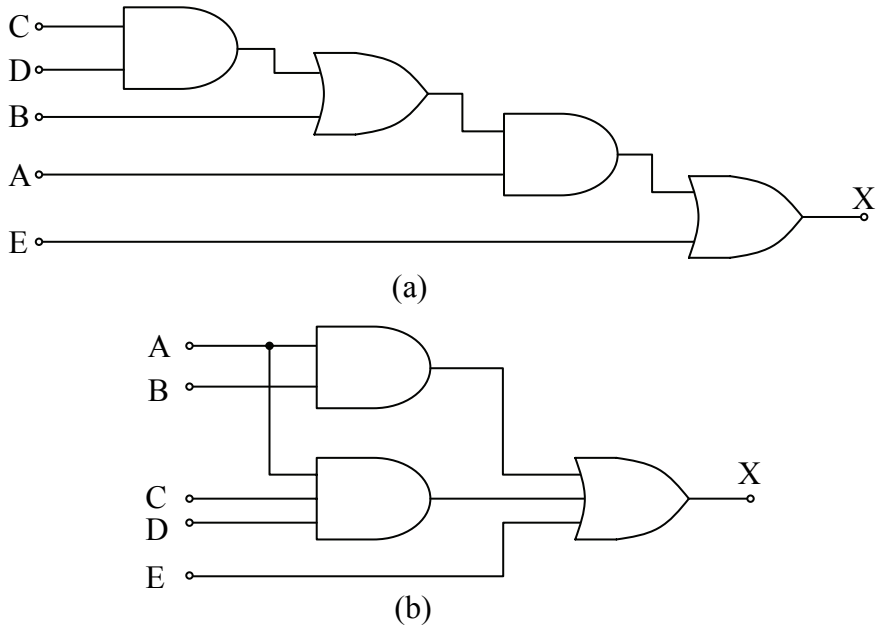



**Fig. 4.1.** Circuit logic levels

All Boolean functions can always be represented as a SOP and, alternatively, as a product of sums (POS). The Karnaugh map (K-map) is a graphical representation of a Boolean function which is particularly useful in determining the minimized (simplest) SOP or POS expressions for a function. The specific arrangement of the K-map eliminates the need for extensive use of Boolean algebra to simplify the equation. Instead, the simplification is done graphically using the K-map.

The K-map contains a *cell* for each input combination. A logic function with n input variables has $2^n$ cells on the K-map. A two-variable K-map has 4 cells, a three variable K-map has 8 cells, and a four-variable K-map has 16 cells. Occasionally, five-variable K-maps are formed by using two four-variable maps. Functions requiring larger K-maps are usually handled by computer simulation and Boolean algebra techniques.

Fig. 4.2 shows two alternative notations for a K-map for 4 variables. They are equivalent. Each square corresponds to one of the possible combinations, or minterms, of the 4 variables. The combination corresponding to any particular square can be determined from the coordinates of the associated row and column. Note that these coordinates are assigned in such a way that, in moving from one square to the next, either horizontally or vertically, only one variable changes from true to complemented, or complemented to true, form. Two squares are defined to be "adjacent" if this condition applies. Note that this property also applies to squares at opposite ends of rows and columns, so that these squares are also adjacent. Similar maps can be formed for any number of variables, though maps for more than 6 variables are generally too unwieldy.

Boolean functions are mapped by first writing the function as a sum of products (it may be necessary to remove parentheses) and then placing a "1" in all squares corresponding to each product term. The map thus contains the same information as the truth table of the function. Consider the function shown in Equation (4.1) and its K-map shown in Fig. 4.3.

$$X = A + \overline{B}\,C + B\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\overline{C}\,D \qquad (4.1)$$

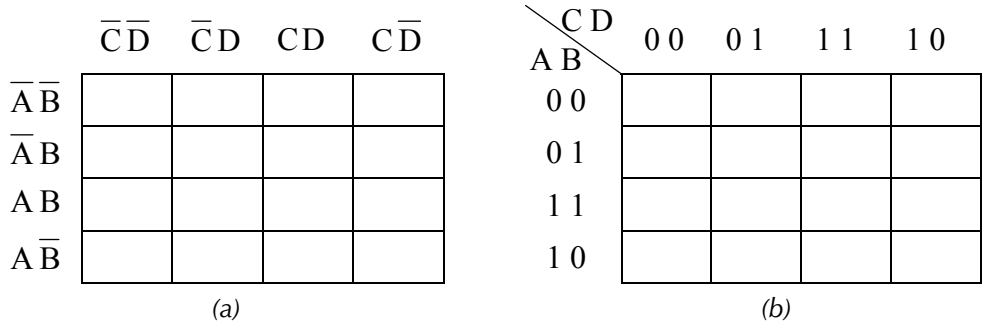



**Fig. 4.2.** Karnaugh maps for 4 variables

| | $\overline{C}\,\overline{D}$ | $\overline{C}D$ | $CD$ | $C\overline{D}$ |
|---|---|---|---|---|
| $\overline{A}\,\overline{B}$ | | 1 | 1 | 1 |
| $\overline{A}\,B$ | 1 | | | |
| $A\,B$ | 1 | 1 | 1 | 1 |
| $A\,\overline{B}$ | 1 | 1 | 1 | 1 |

**Fig. 4.3**. K-map of Equation (4.1)

The way in which the various terms map onto the squares should be noted. The single term, A, maps onto the 8 squares in the lower half on the map which correspond to A = 1. The 2-variable term maps onto 4 squares (only two of which are new in this case), the 3-variable term maps onto 2 squares, and the 4-variable term maps onto a single square.

The procedure for finding the simplest expression for a given K-map is the reverse of the mapping process described above. Adjacent squares are combined so that they can be represented by a single term. An expression is minimized if it has the fewest possible number of terms and each term has the fewest possible number of variables. Thus, in deriving minimized expressions from a given K-map, all function 1's should be included by making the fewest number of combinations in which each combination includes the maximum number of squares.

| | $\overline{C}\,\overline{D}$ | $\overline{C}D$ | $CD$ | $C\overline{D}$ |
|---|---|---|---|---|
| $\overline{A}\,\overline{B}$ | | | | 1 |
| $\overline{A}B$ | | 1 | | 1 |
| $AB$ | 1 | | | 1 |
| $A\overline{B}$ | 1 | | | 1 |

*(a) Function X*

| | $\overline{C}\,\overline{D}$ | $\overline{C}D$ | $CD$ | $C\overline{D}$ |
|---|---|---|---|---|
| $\overline{A}\,\overline{B}$ | | 1 | 1 | |
| $\overline{A}B$ | 1 | | | 1 |
| $AB$ | 1 | | | 1 |
| $A\overline{B}$ | | 1 | 1 | |

*(b) Function Y*

**Fig. 4.4**. Simplifiable K-maps

As an example, consider the function mapped in Fig. 4.4 (a). The column of 1's on the right can all be combined, the two 1's in the lower left column can be combined with the 1's on the opposite ends of the rows, and the remaining 1 has no adjacent squares so that it cannot be combined. The minimized functional expression is shown in (4.2).

$$X = C\overline{D} + A\overline{D} + \overline{A}B\overline{C}D \qquad (4.2)$$

Equation (4.2) is the minimized 2-level SOP expression for the function, X. It should be noted that any SOP or POS expression can be implemented by a 2-level circuit; input inverters necessary to generate variable complements are ignored in determining level since, in many practical applications, variables are available in both true and complemented form.

To find the minimized POS form for a function it is simply necessary to find the minimized SOP form for the function complement and then complement

the result. Consider the function, Y, shown in Fig. 4.4 (b). Examining the empty squares (these would be 1 for the complement of the function), it can be seen that 2, 4-square combinations can be made (center and corners). The resulting expression and corresponding minimized POS form are shown in Equation (4.3).

$$\overline{Y} = BD + \overline{B}\,\overline{D}$$

$$Y = \overline{BD + \overline{B}\,\overline{D}} = (\overline{B} + \overline{D})(B + D) \qquad (4.3)$$

## LABORATORY ASSIGNMENTS

### Assignment 4.1

Construct the K-map for the function given in Equation (4.4). Determine the minimized SOP expression and design the corresponding circuit using NAND gates. Breadboard the circuit and measure its truth table to confirm correct operation.

*Hint*: Find a SOP expression for the function by removing the parentheses before trying to map. The circuit can be implemented with a single 7400 chip.

$$X = [A(\overline{B}C + D) + \overline{A}C]D \qquad (4.4)$$

### Assignment 4.2

Using the map constructed in Assignment 4.1, determine the minimized POS expression for the function of Equation (4.4), by complementing the minimized SOP expression for the function complement. Build the corresponding circuit using NOR gates and measure its truth table to verify correct operation.

*Hint:* The circuit can be implemented with a single 7402 chip.

### Assignment 4.3

Find the minimized SOP expression for the function shown in Fig. 4.5 (a). Construct the corresponding 2-level circuit using NAND gates and inverters. Measure and record its truth table to verify correct operation.

### Assignment 4.4

Determine both minimized SOP and POS expressions for the map shown in Fig. 4.5 (b). There are two equally simple minimized expressions. Design circuits to implement both expressions using only 2-input NAND gates and inverters.

Construct the simplest of these two circuits. Measure and record its truth table to verify correct operation.

*Hint* Factor the minimized SOP expressions to avoid using a 3-input gate. The best circuit will be a 3-level circuit, which can be built with one inverter and one 7400 chip.

| | $\overline{C}\,\overline{D}$ | $\overline{C}D$ | $CD$ | $C\overline{D}$ |
|---|---|---|---|---|
| $\overline{A}\,\overline{B}$ | | | | 1 |
| $\overline{A}B$ | | | 1 | 1 |
| $A\,B$ | 1 | 1 | | |
| $A\,\overline{B}$ | 1 | 1 | | 1 |

*(a) Function X*

| | $\overline{C}\,\overline{D}$ | $\overline{C}D$ | $CD$ | $C\overline{D}$ |
|---|---|---|---|---|
| $\overline{A}\,\overline{B}$ | 1 | 1 | | 1 |
| $\overline{A}B$ | | | 1 | 1 |
| $A\,B$ | | | 1 | 1 |
| $A\,\overline{B}$ | 1 | 1 | | 1 |

*(b) Function Y*

**Fig. 4.5**. K-maps for Assignment 4.3

*In the space provided below, determine minimized SOP or/and POS expressions, and design corresponding circuit for the given assignment using required gates.*

**Assignment No ___ .**

**K-map:**

| | $\overline{C}\,\overline{D}$ | $\overline{C}D$ | $CD$ | $C\overline{D}$ |
|---|---|---|---|---|
| $\overline{A}\,\overline{B}$ | | | | |
| $\overline{A}B$ | | | | |
| $A\,B$ | | | | |
| $A\,\overline{B}$ | | | | |

**Minimized SOP or/and POS expressions:**

**Designed logic circuit:**

**Truth Table:**

| A | B | C | D | Output |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

EXPERIMENT

# 5

# CODE CONVERTERS

## OBJECTIVES

1. Understand unit distance codes; design circuitry to convert between Gray code and binary.
2. Exercising the design of combinational circuit to convert between different BCD codes.
3. Use 7-segment displays and 7-segment decoder integrated circuits to design BCD display circuitry.

## EQUIPMENT REQUIRED

Digital Logic Trainer
Digital Multimeter (DMM)
Integrated circuits (See Assignments)

## BASIC INFORMATION

Code conversion is frequently required in digital system design, particularly at the interface between the internal circuitry and the devices to which the system is connected. For example, many digital systems require that some output be displayed using decimal numbers so that it is readily understood. In this case, output quantities are most conveniently represented in BCD in order to drive commonly used display devices. Another example is provided by digital systems, typically in a control application, that receive input from sensors which encode the position of a rotating shaft in such a way as to minimize errors. In both of these cases, it is necessary to design circuitry to convert between the codes used by the interface devices and the binary number system used internally in the system.

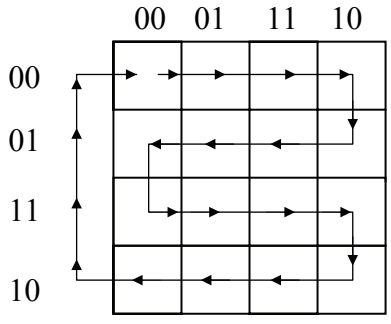
Consider the problem of encoding the position of a rotating shaft. If shaft position were encoded using the normal binary number system then, as the shaft position changed, the binary numbers output from the encoder would

change. Assuming a 4-bit representation, the situation might be encountered in which the output changed from 0111 to 1000. Notice that all four bits would have to change for one increment in position. Since it is impossible for all bits to change simultaneously, there would be some periods of time when the encoder output would be in error, for example 1111 or 1110. This could introduce other errors in the system using this data input. To avoid the problem what is needed is a binary representation, or code, in which only one bit changes at a time as quantity is increased. Such a code is called a unit distance code.

Fig. 5.1 shows a commonly used unit distance code, the Gray code. Although this code is shown for four bits it can be extended to any number of bits for more resolution. Fig. 5.1 (a) shows the truth table for the code while (b) illustrates its relation to the K-map for four variables.

| Decimal | Binary | Gray |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

*(a) Truth table*




*(b) K-map generation*

**Fig. 5.1**. Gray code

It should be noted that K-maps provide a unit distance representation of Boolean functions; moving from any square to the next, either vertically or horizontally or at opposite ends of rows or columns, causes only one bit to change. The Gray code follows the path shown in the above figure beginning with 0000.

One of the reasons that Gray code is commonly used is that it is easy to convert between binary and Gray code representations. It can be seen from the examination of the truth table in Fig. 5.1 (a) that the most significant bits of both codes are the same. In the case of converting from binary to Gray, bit N of the Gray code can be generated by the exclusive-OR of binary bits N and N + 1. The relationship for converting from Gray to binary is somewhat more difficult to see, but careful examination of the truth table shows that binary bit N can be generated by the exclusive-OR of Gray bit N with all more significant bits. Hence, circuitry to convert between 4-bit binary and Gray code would be designed as shown in Fig. 5.2. It should be noted that this algorithm can be extended to binary and Gray codes of any length.

Conversion between binary and BCD is more difficult, but MSI chips are available to use in this process. The 74184 and 74185 are integrated circuits designed for conversion between binary and BCD. Both chips are custom read-only memory (ROM) chips that have been programmed to accomplish BCD to binary conversion (74184) and binary to BCD conversion (74185).

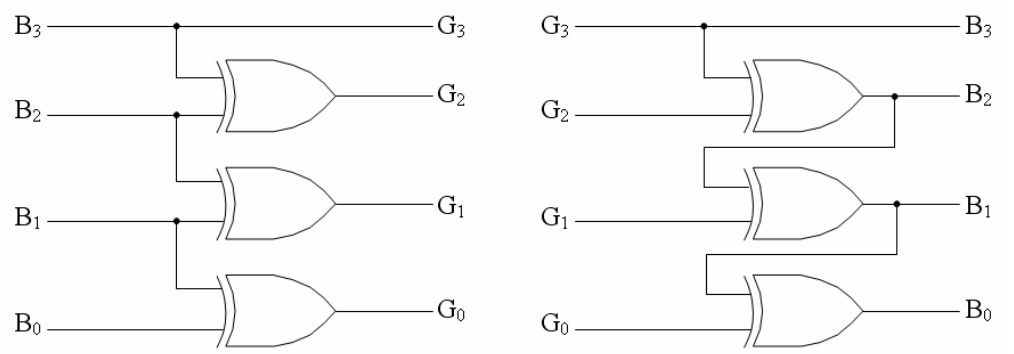
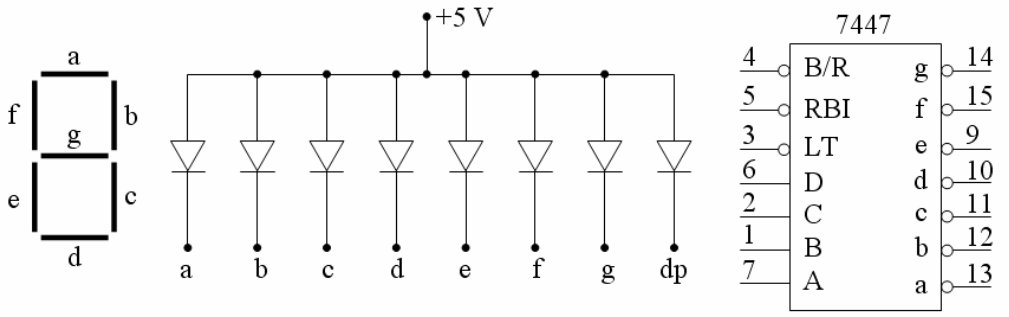



**Fig. 5.2.** Binary and Gray code conversion

The last code conversion application to be discussed is conversion between BCD and 7-segment code. Light emitting diode (LED) display devices are available in DIP packages to permit a visual display of the ten decimal digits. Eight LEDs are used, connected either in a common anode or common cathode arrangement, to display the decimal point and the seven individual segments. The decimal digits are displayed by selectively forward biasing the segment diodes to create appropriate illuminated patterns. In the common anode device, diodes are forward biased by applying a LOW level at the segment inputs. This type of display is illustrated in Fig. 5.3 (a). To display a

decimal digit it is thus necessary to have a circuit which will accept a 4-bit BCD input and produce seven outputs which will cause the required segments to be illuminated. An integrated circuit frequently used for common anode display devices is the 7447 decoder/driver, the logic symbol for which is shown in Fig. 5.3 (b).




*(a) 7-segment LED display device* *(b) Decoder/driver*

**Fig. 5.3.** Common anode LED display device

The BCD input is applied to pins, D to A, with D being the most significant bit. The segment outputs, *a* to *f*, are active LOW so that they can forward bias the corresponding diode to which they are connected. It should be noted that the 7447 uses open collector outputs and it is necessary to connect the 7447 outputs to the LED inputs through a current limiting resistor. For the LED devices used in this laboratory, 5 to 10 mA forward current is required for adequate light output and the specified forward voltage drop is 1.6 V. Therefore, a series current-limiting resistors from 270 Ω to 680 Ω should be used.

In addition to the BCD inputs and 7-segment code outputs, three additional pins are available on the 7447 to permit testing the display and expanding the circuitry to drive multidigit displays with leading and/or trailing zero blanking. When asserted, the active low LT input will turn on all outputs and their corresponding LEDs. The B/R pin functions as both an input and an output. If it used as an input and forced LOW, all outputs are off regardless of the state of any other input. This pin functions as an output in response to the ripple blanking input, RBI. If this pin is asserted active LOW, and the BCD input is zero, the B/R output will be forced LOW and all segment outputs will be off. This property makes it possible to blank either leading or trailing zeros in a multidigit display by connecting the B/R output to the RBI input of adjacent digit drivers.

## LABORATORY ASSIGNMENTS

### Assignment 5.1

Design, build and test circuits for converting 4-bit Gray code to binary and 4-bit binary to Gray code using XOR gates and the algorithm described in the Basic Information. (This can be done with one 7486 IC). Connect the circuit to four switches and four indicator lamps and check for proper operation using a truth table shown in Fig. 5.1, (a).

### Assignment 5.2

The circuit shown in the Basic Information for converting Gray code to binary is a multilevel circuit with propagation delays that become increasingly more severe as the number of bits increases. Design a 2-level circuit using only NAND gates and inverters that will convert a 3-bit Gray code to a 3-bit binary output (Table 5.1). Use any of the gates 7400, 7402, 7404, 7408, 7410, 7420, 7432 and 7486, but minimize the total number of ICs used. Breadboard the circuit and test its operation for all input conditions.

*Hint:* Construct K-maps for the three binary output bit functions.

**Table 5.1**

| Decimal | Gray | | | Binary | | |
|---|---|---|---|---|---|---|
| | $G_3$ | $G_2$ | $G_1$ | $B_3$ | $B_2$ | $B_1$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 0 | 0 | 1 | 1 |
| 4 | 1 | 1 | 0 | 1 | 0 | 0 |
| 5 | 1 | 1 | 1 | 1 | 0 | 1 |
| 6 | 1 | 0 | 1 | 1 | 1 | 0 |
| 7 | 1 | 0 | 0 | 1 | 1 | 1 |

**Table 5.2**

| Decimal | 8421 Code | | | | 2421 Code | | | |
|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | a | b | c | d |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

### Assignment 5.3

Design a combinational circuit with four input lines that represent a decimal digit in BCD (8421) and four output lines that generate self-complementing code 2421 (Table 5.2). Provide a fifth output that detects an error in the input BCD number. This output should be equal to logic-1 when the four inputs

have one of the unused combinations of the BCD code. Use any of the gates 7400, 7402, 7404, 7408, 7410, 7420, 7432 and 7486, but minimize the total number of ICs used.

**Assignment 5.4**

a) Test circuit for converting BCD to seven-segment code using the 7447 decoder driver (Fig. 5.4). Use binary switches (on the left side of the Y-0039 Digital Logic Trainer) to connect the four BCD inputs (A1, B1, C1, D1 and A2, B2, C2, D2) to logic 1 or logic 0. Record in Table 5.3 the number you would expect to see on the display if each code were placed on the 7447 inputs.

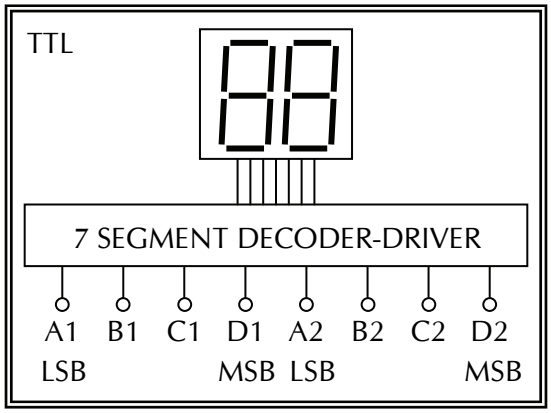



**Fig. 5.4.** Y-0039 2x7 Segment Decoder-Driver Section

**Table 5.3.** Decoder/Driver Observations

| BCD Code | | Number Expected | | Number Observed | |
|---|---|---|---|---|---|
| A1 B1 C1 D1 | A2 B2 C2 D2 | First | Second | First | Second |
| 0000 | 0000 | | | | |
| 0010 | 0001 | | | | |
| 0011 | 0101 | | | | |
| 0100 | 0110 | | | | |
| 0111 | 1000 | | | | |
| 1001 | 1010 | | | | |
| 1011 | 1100 | | | | |
| 1101 | 1111 | | | | |

b) Test circuit for converting BCD to seven-segment code using the 7447 decoder driver (Fig. 5.5). Use binary switches (on the left side of the Y-0039 Digital Logic Trainer) to connect the four BCD inputs (A, B, C, D) to logic 1 or logic 0. Record in Table 5.3 the number you would expect to see on the display (HDSP 5501) if each code were placed on the 7447 inputs. When a BCD input is connected, measure the seven-segment outputs, *a* to *g*, with a DMM. Record the measurements as high or low in Table 5.4. Also record the number displayed on the LED.

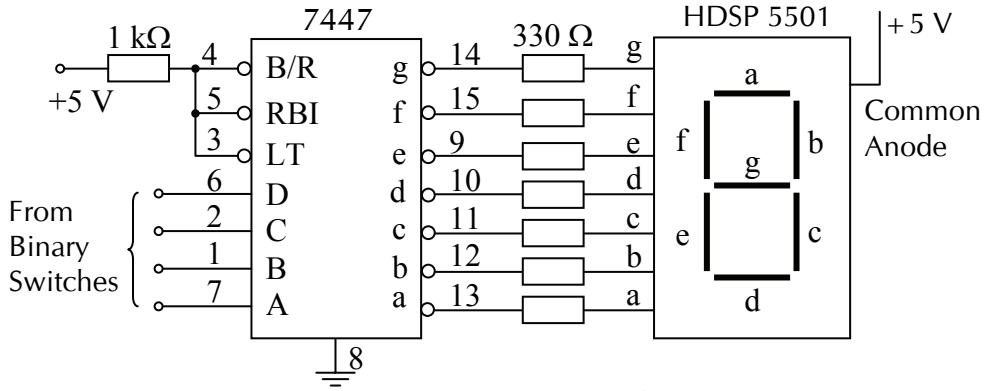



**Fig. 5.5.** Experimental circuit

**Table 5.4.** Decoder/Driver Measurements

| BCD Code | Number Expected | Number Observed | Outputs | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | *a* | *b* | *c* | *d* | *e* | *f* | *g* |
| 0000 | | | | | | | | | |
| 0010 | | | | | | | | | |
| 0100 | | | | | | | | | |
| 0110 | | | | | | | | | |
| 1000 | | | | | | | | | |
| 1001 | | | | | | | | | |
| 1100 | | | | | | | | | |
| 1111 | | | | | | | | | |

*In the space provided below, determine minimized expressions, and design corresponding circuit for the assignments 5.2 and 5.3 using required gates.*

**Assignment No ___.**

**K-maps and minimized Boolean expressions:**

**Designed logic circuit:**

EXPERIMENT

# 6

# DESIGN WITH MULTIPLEXERS AND DEMULTIPLEXERS

## OBJECTIVES

1. Understand the operation of integrated circuits used for multiplexing and demultiplexing.
2. Design multiplexing and demultiplexing circuitry using appropriate integrated circuits.

## EQUIPMENT REQUIRED

Digital Logic Trainer

Integrated circuits (See Assignments)

## BASIC INFORMATION

There are many applications where it is required to use one wire or other transmission path to carry signals from two or more sources. This process is called multiplexing and the circuits which accomplish this are called multiplexer (MUX) circuits. Essentially, a multiplexer circuit is an electronic switch which has the capability of connecting one line from a number of input lines to an output line as determined by the binary input values supplied to a few select lines. A number of chips are available to implement the more commonly encountered multiplexer requirements. They vary from a 2 to 1 multiplexer (four in a package) to a 16 to 1 multiplexer.

All of these commonly used integrated circuits have a number of features in common. Each has an active LOW enable (strobe) input(s), the purpose of which is to allow for cascading devices.The remaining pins are divided between data inputs, select lines and outputs. The binary code impressed on the select inputs determines which input line is connected to the output line.

The 74157 is a quad arrangement of four identical 2 to 1 stages which share a common select line (Fig. 6.1). Since input selection is from 1 to 2 lines for each stage, only a single select line is required. For this device, the A inputs

are selected for each stage when S = 0 (input $\overline{A}B$). Note that the outputs are available in true form for all four stages.

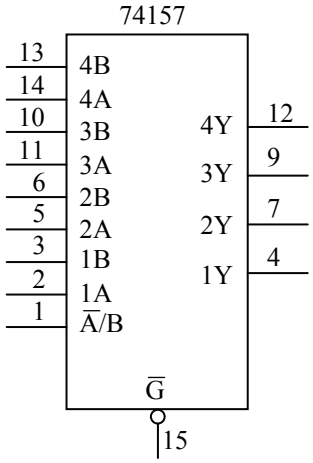



**Fig. 6.1.** 74157 quad 2:1 IC multiplexer

74153 is a dual 4 line-to-1 line multiplexer. It has the schematic representation shown in Fig 6.2. Selection lines S1 and S0 select the particular input to be multiplexed and applied to the output.

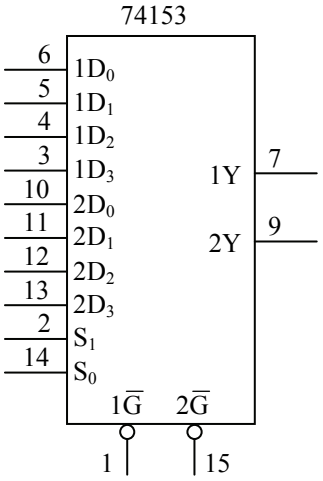



**Fig. 6.2.** 74153 dual 4:1 IC multiplexer

Each of the strobe signals acts as an enable signal for the corresponding multiplexer.

Table 6.1. shows the multiplex function of 74153 in terms of select lines. Note that each of the on-chip multiplexers act independently from the other, while sharing the same select lines S1 and S0.

**Table 6.1.** 74153 truth table

| Multiplexer 1 | | | |
|---|---|---|---|
| Strobe | Select lines | | Output |
| 1G | S1 | S0 | 1Y |
| 1 | X | X | 0 |
| 0 | 0 | 0 | 1D0 |
| 0 | 0 | 1 | 1D1 |
| 0 | 1 | 0 | 1D2 |
| 0 | 1 | 1 | 1D3 |

| Multiplexer 2 | | | |
|---|---|---|---|
| Strobe | Select lines | | Output |
| 2G | S1 | S0 | 2Y |
| 1 | X | X | 0 |
| 0 | 0 | 0 | 2D0 |
| 0 | 0 | 1 | 2D1 |
| 0 | 1 | 0 | 2D2 |
| 0 | 1 | 1 | 2D3 |

The 74151, an 8 to 1 MUX shown in Fig. 6.3, functions in an identical manner except both true and complement outputs are available. Selection lines S2, S1 and S0 select the particular input to be multiplexed and applied to the output. Strobe S acts as an enable signal. Table 6.2 shows the multiplex function of 74151 in terms of select lines.

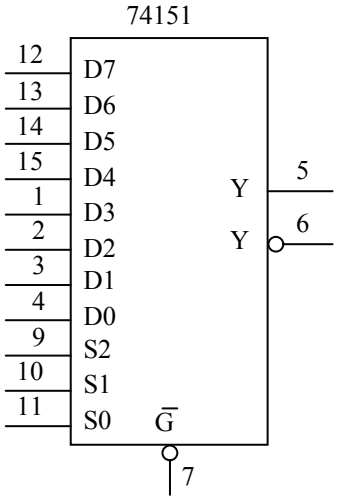



**Fig. 6.3.** 74151 8:1 IC multiplexer

**Table 6.2.** 74151 truth table

| Strobe | Select lines | | | Output |
|---|---|---|---|---|
| G | $S_2$ | $S_1$ | $S_0$ | Y |
| 1 | X | X | X | 0 |
| 0 | 0 | 0 | 0 | D0 |
| 0 | 0 | 0 | 1 | D1 |
| 0 | 0 | 1 | 0 | D2 |
| 0 | 0 | 1 | 1 | D3 |
| 0 | 1 | 0 | 0 | D4 |
| 0 | 1 | 0 | 1 | D5 |
| 0 | 1 | 1 | 0 | D6 |
| 0 | 1 | 1 | 1 | D7 |

The 74150 is a 16 to 1 MUX which therefore requires four select lines to select one of the 16 inputs. Note that the output, Y, is inverted for this chip so that the output in this case would be the complement of the level applied to D9.

No experiments using 74150 are assigned in this manual, so its schematic diagram and truth table are not presented as well.

Multiplexers are versatile circuits that are used in applications such as data selection, cascaded operation, binary word multiplexing, time-division multiplexing, and logic function generation.

Using multiplexing circuitry allows a single line or small set of lines to be used to carry multiple signals. Demultiplexing represents the reverse process. A demultiplexer (DMUX) circuit has the capability of taking data on a single line or small set of lines, and connecting that data selectively to one of a larger number of output lines. The integrated circuit decoders previously studied can accomplish this task. For this reason these integrated circuits are often called decoder/demultiplexer circuits.

Consider the circuit shown in Fig. 6.4, which uses the 74138 3-bit decoder. Assume that eight separate logic signals have been multiplexed onto the single data input lines. Notice that the data input has been connected to the active LOW enable, E2. If a select code is applied to the normal decoder data input lines, the corresponding output line will be LOW if the chip is enabled (if the chip is not enabled recall that all output lines remain HIGH). This circuit would function in a similar way if the data input line were connected to the active HIGH enable, E3, except the data at the output line would be the inverse of the data at the input. Other decoder chips can be used in a similar manner as demultiplexers. For example, the 74154 4-bit binary decoder can be used to demultiplex a single line onto 1 of 16 outputs.

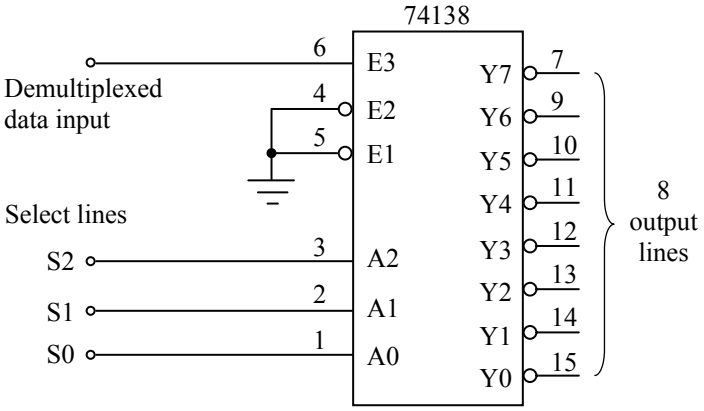



**Fig. 6.4**. 74138 1 to 8 DMUX circuit

Table 6.3 shows the demultiplex function of 74138 in terms of select lines.

**Table 6.3.** 74138 truth table

| Inputs | | | | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Enable | | | Select | | | | | | | | | | |
| E3 | E2 | E1 | $A_2$ | $A_1$ | $A_0$ | Y0 | Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 |
| 0 | X | X | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X | 1 | X | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X | X | 1 | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Demultiplexer circuits are multipurpose AND-array circuits used extensively in data transmission and in data routing applications.

## LABORATORY ASSIGNMENTS

### Assignment 6.1

Design a multiplexer which will multiplex one of two nibbles (4 bits) of data to the output of the multiplexer. Use one 74157 integrated circuit.

*Hint:* For this device, the A inputs are selected for each stage when S = 0. The nibble 1 ($S_0$ - $S_3$) is entered into the A inputs of 74157, and the nibble 2 ($R_0$ - $R_3$) is entered into the B inputs of 74157 (Fig. 6.5).

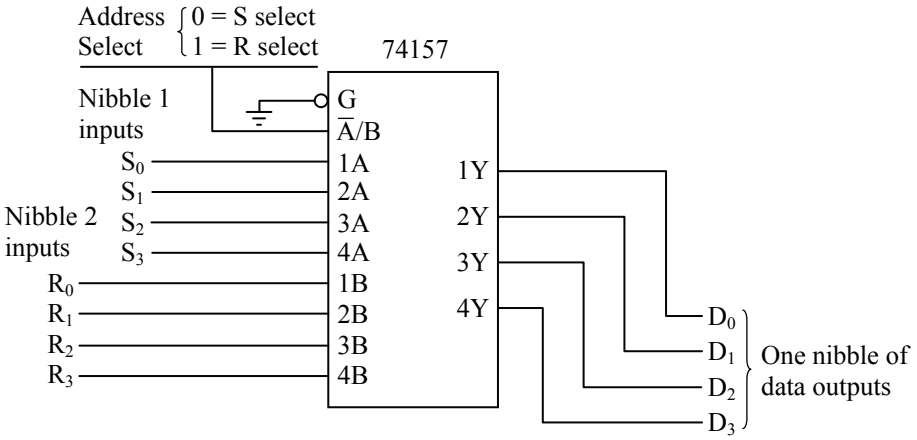



**Fig. 6.5.** Nibble multiplexing

1. Construct the circuit and write the pin numbers on the designed circuit.
2. Place the 74157 chip on a breadboard and assemble the connections to implement the circuit illustrated in Fig. 6.5.
3. Apply power to the circuit ($V_{CC}$ – pin 16, GND – pin 8).
4. Test the circuit using appropriate data points.

### Assignment 6.2

Design a multiplexer which will multiplex one of two bytes of data to the output of the multiplexer. Use two 74157 integrated circuits. Construct the circuit and test it using appropriate data points.

*Hint:* The select inputs and enable inputs on the 74157s are tied common and controlled by the same select and enable inputs, respectively. The lower nibble of each byte is entered into one 74157, and the upper nibble of each byte is entered into the other 74157 (Fig. 6.6).

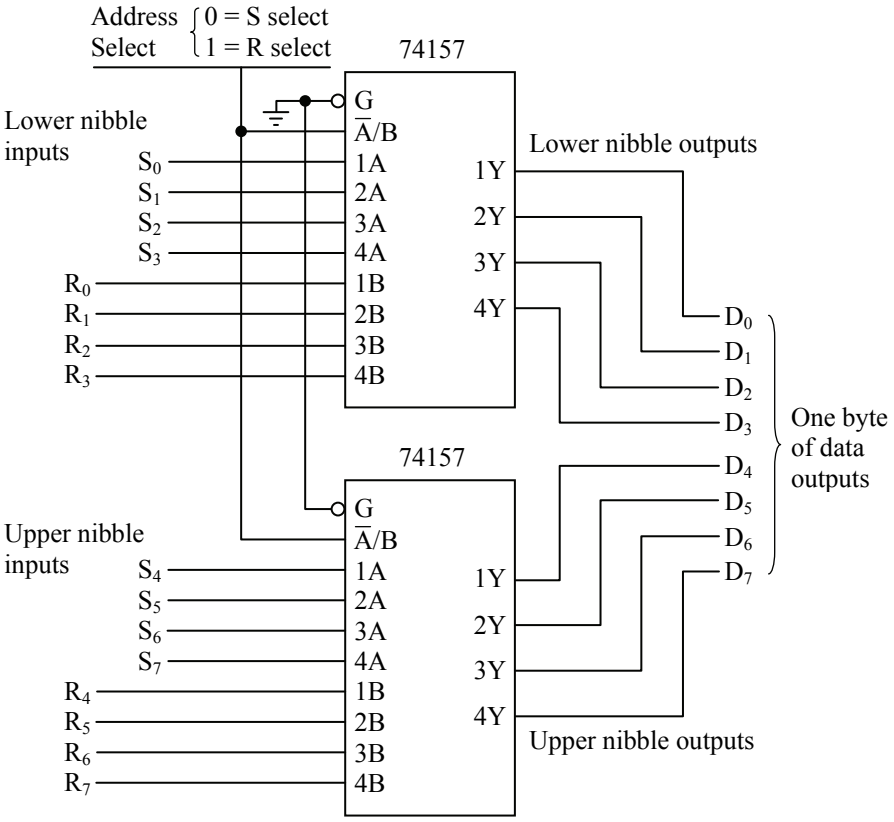



**Fig. 6.6.** Byte multiplexing

1. To simulate signals for S inputs ($S_0$ - $S_7$) use outputs from the TTL Function Switches of the Digital Logic Trainer Y-0039 and for R inputs ($R_0$ - $R_7$) use output signals from 8 NOT gates of two 7404 IC connecting inputs of NOT gates to + 5 V or to ground (depending on the required input code $R_0$ - $R_7$).
2. Construct the circuit and write the pin numbers on the designed circuit.
3. Place the 74157 and 7404 chips on a breadboard and assemble the connections to implement the circuit illustrated in Fig. 6.6.
4. Apply power to the circuit ($V_{CC}$ – pin 16, GND – pin 8 for 74157 and $V_{CC}$ – pin 14, GND – pin 7 for 7404).
5. Test the completed circuit using appropriate data points.

**Assignment 6.3**

The **digital multiplexer** can be used to implement sum-of-products (SOP) expressions (to provide a logic function generation) and its ouiput can represent the Boolean expression for any combinational logic function.

**a)** Design, construct, and test a circuit which uses an 74153 to implement a sum-of-products expression:

$$Y = f(A,B,C) = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + A\,B\overline{C} + A\,BC$$

The multiplexer implementation table is shown in Table 6.4.

**Table 6.4.** MUX implementation table

| | Connect A and B to select lines | | | Express F in terms of the other input (C) | |
|---|---|---|---|---|---|
| | A | B | C | F | |
| MUX Input 0 | 0 | 0 | 0 | 1 | $F = \overline{C}$ (So connect $\overline{C}$ to MUX input 0) |
| | 0 | 0 | 1 | 0 | |
| | 0 | 1 | 0 | 0 | F = C |
| | 0 | 1 | 1 | 1 | |
| | 1 | 0 | 0 | 0 | F = 0 |
| | 1 | 0 | 1 | 0 | |
| | 1 | 1 | 0 | 1 | F = 1 |
| | 1 | 1 | 1 | 1 | |

The circuit can be implemented as shown in Fig. 6.7.

Keep in mind in the example above that bits A and B were connected to the select lines. If any other bits are connected to the select lines, then the implementation table needs to be rearranged.

1. Construct the circuit and write the pin numbers on the designed circuit.
2. Place the 74153 chip on a breadboard and assemble the connections to implement the circuit illustrated in Fig. 6.7.
3. Apply power to the circuit ($V_{CC}$ – pin 16, GND – pin 8).
4. Test the completed circuit using appropriate data points.

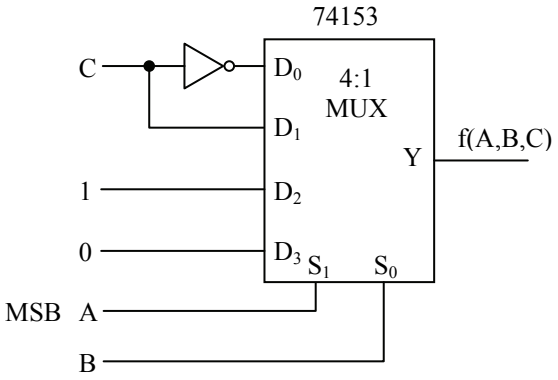



**Fig. 6.7.** f(A,B,C) implemented using 4:1 multiplexer

**b)** Design, construct, and test a circuit which uses an 74151 to implement the logic function specified in the truth table (Table 6.5). Compare this method with a discrete logic gate implementation.

To use a multiplexer as a logic function generator, the logic function is formed by srtting the data inputs to the appropriate logic level. The select inputs become the inputs for the variables specified in the function.

**Table 6.5.** Logic function implementation table

| Inputs | | | Output |
|---|---|---|---|
| R | S | T | M |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

Notice from the truth table that M is a 1 for the following input variable combinations: 000, 010, 011, 100, 101. For all other combinations, M is 0. For this function to be implemented with the multiplexer, the data input selected by each of the above-mentioned combinations must be connected to a HIGH ($V_{CC}$) through pull-up resistor. All other data inputs must be connected to a LOW (ground) as shown in Fig. 6.8. The logic function is described as

$$M = \overline{R}\,\overline{S}\,\overline{T} + \overline{R}\,S\,\overline{T} + \overline{R}\,S\,T + R\,\overline{S}\,\overline{T} + R\,\overline{S}\,T$$

The implementation of this function with logic gates would require five 3-input AND gates, one 5-input OR gate, and three inverters unless the expression can be simplified.

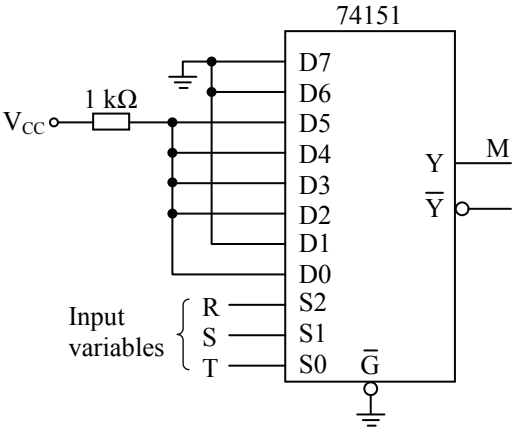



**Fig. 6.8.** Three variable logic function generator

1. Construct the circuit and write the pin numbers on the designed circuit.
2. Place the 74151 chip on a breadboard and assemble the connections to implement the circuit illustrated in Fig. 6.8.
3. Apply power to the circuit ($V_{CC}$ – pin 16, GND – pin 8).
4. Test the circuit using appropriate data points.

**c)** Consider the following sum-of-products expression:

$$Y = f(A,B,C) = A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B} + \overline{B}\,\overline{C}$$

To implement this expression with a multiplexer, it must first be expanded into each of its unique terms. It is advantageous to express the final argument in summation form:

$$Y = f(A,B,C) = A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}(C + \overline{C}) + \overline{B}\,\overline{C}(A + \overline{A})$$
$$Y = f(A,B,C) = A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C}$$

The implementation of this function is shown in Fig. 6.9. Note that the inputs to the multiplexer that are identified as TRUE (logic 1) in the summation expression are tied to a logic 1. The remaining inputs are tied to a logic 0.

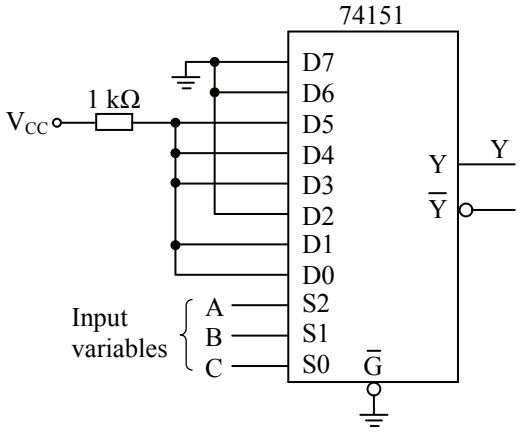



**Fig. 6.9.** Three variable logic function generator

1. Construct the circuit and write the pin numbers on the designed circuit.
2. Place the 74151 chip on a breadboard and assemble the connections to implement the circuit illustrated in Fig. 6.9.
3. Apply power to the circuit ($V_{CC}$ – pin 16, GND – pin 8).
4. Test the circuit using appropriate data points.

**Assignment 6.4**

Design, construct, and test a circuit using an 74151 that will convert 8 bits of parallel data to a serial stream of data on the output. Use an LED to signal a logic 1 output on the output line.

1. Construct the circuit and write the pin numbers on the designed circuit.
2. Place the 74151 chip on a breadboard and assemble the connections to implement the circuit illustrated in Fig. 6.10.
3. Connect any data (logic 1's or 0's) to the inputs (D0 - D7) of the 74151. You must simulate a counter circuit on the A, B, and C inputs to select each bit of data to be placed on the output line
4. Apply power to the circuit ($V_{CC}$ – pin 16, GND – pin 8).
5. Stimulate inputs A, B, and C, and data inputs D0 through D7 to demonstrate circuit operation.

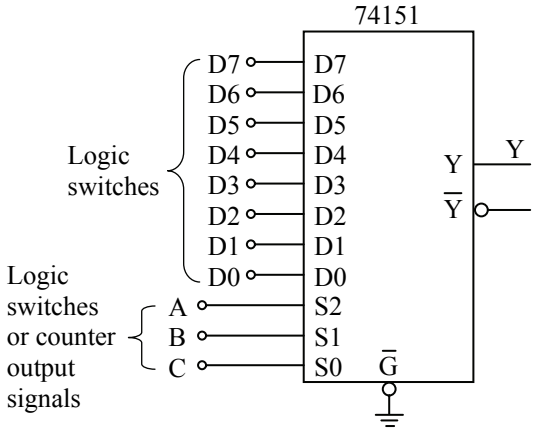



**Fig. 6.10.** Conversion of parallel data to a serial stream

**Assignment 6.5**

Using a 74151 and a 74138, design a multiplexer/demultiplexer circuit which will take 8 data lines, multiplex them onto a single output line, and then demultiplex the single data line onto 8 output lines (Fig.6.11). The selection of the input and output data lines must be controlled by a single set of logic signals, A, B, C, such that corresponding input and output lines are always selected. For example, setting A, B, C, to 011 must select input line 3 and output line 3. Build the circuit and test it by applying a pulse source to each input line and confirming its transmission to the correct output line when the appropriate code is input to the select lines.

1. Construct the circuit and write the pin numbers on the designed circuit.
2. Place the 74151 and 74138 chips on a breadboard and assemble the connections to implement the circuit illustrated in Fig. 6.11.
3. Apply power to the circuit ($V_{CC}$ – pin 16, GND – pin 8 for both 74151 and 74138).
4. Test the circuit using appropriate data points.

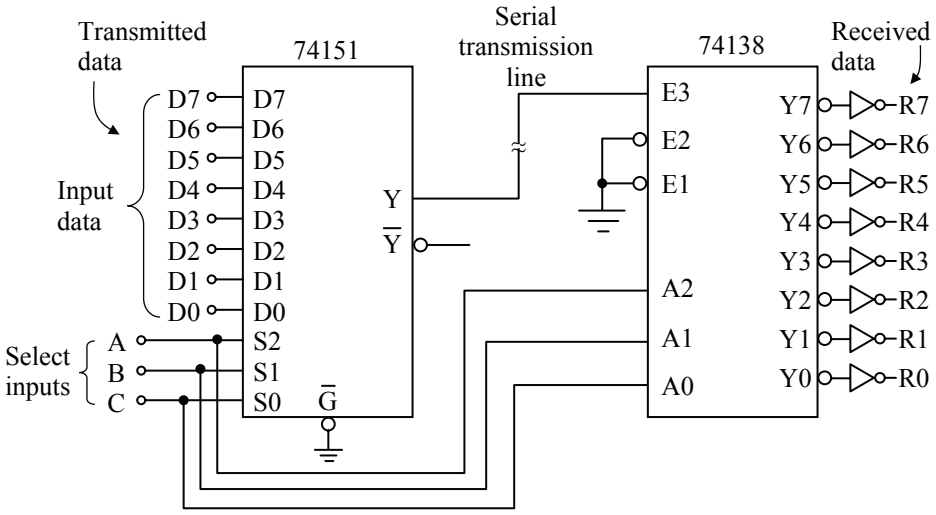



**Fig. 6.11.** Multiplexer/demultiplexer system

**Assignment 6.6**

Design, construct, and test a circuit which uses an 74138 demultiplexer to implement a sum-of-products expression.

$$Y = f(A, B, C) = \overline{A}BC + B\overline{C}$$

1. Convert the expression to summation form:

2. The demultiplexer output is selected, and will go low, by the address on inputs A, B, and C when the IC is enabled. Therefore, we can create the output function Y by summing together the outputs indicated by the summation form of expression obtained in step 1. Since the outputs of the demultiplexer are active-low, this is done with a NAND gate. Connect each of the TRUE term outputs of the demultiplexer in Fig. 6.12 (indicated by the summation equation) to an input of the NAND gate. Connect all unused NAND inputs to a logic 1
3. Fill in the column labeled Y of Table 6.6 with the anticipated output logic of the 1 to 8 demultiplexer/NAND circuit configured to generate the function of above expression.

4. Write the pin numbers on the designed circuit.
5. Place an 74138/7420 chip pair on a breadboard and assemble the connections to implement the completed circuit illustrated in Fig. 6.12.
6. Apply power to the circuit ($V_{CC}$ – pin 16, GND – pin 8).
7. Stimulate inputs A, B, and C, to complete the $\overline{Y}$ column of Table 6.6.

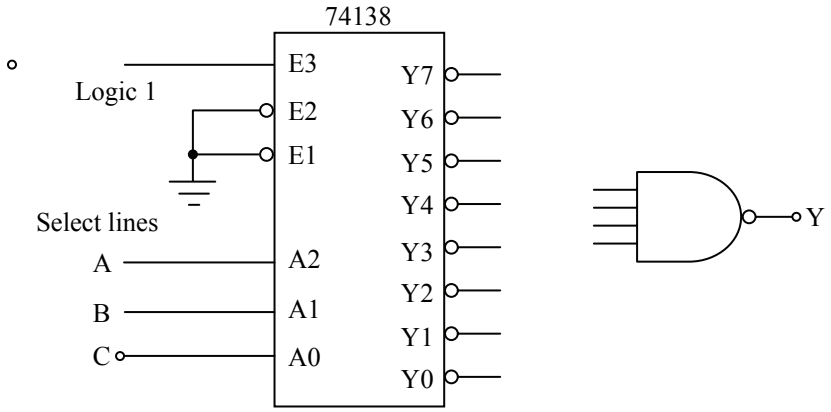



**Fig. 6.12**. SOP expression implementation using 74138 DMUX circuit

**Table 6.6.** Demultiplexer/NAND circuit implementation table

| Inputs | | | Demultiplexer output | NAND output |
|---|---|---|---|---|
| A | B | C | | |
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

*In the space provided below, draw the corresponding designed circuit and record the test results.*

**Assignment No ___.**

**Designed circuit:**

**Test results:**

EXPERIMENT

# 7

# ARITHMETIC CIRCUITS

## OBJECTIVES

1. Investigate the operation of a half and full adders.
2. Design circuits using typical medium scale integrated arithmetic circuits.

## EQUIPMENT REQUIRED

Digital Logic Trainer

Integrated circuits (See Assignments)

## BASIC INFORMATION

The fundamental binary arithmetic operation is the addition of two bits. Consider adding two bits, A and B. The addition process produces two results: a sum bit, S, and a carry bit, C. From the rules for binary addition, the carry bit is 1 if A = B = 1, while the sum bit is 1 if (A, B) = (1, 0) or (0, 1), i.e., if A and B are different. This basic operation is described algebraically in equations (7.1). A circuit, which performs this operation, is called a "half adder". One possible implementation is shown in Fig. 7.1.

$$S = \overline{A}\,B + A\,\overline{B} = A \oplus B$$
$$C = A\,B \qquad (7.1)$$

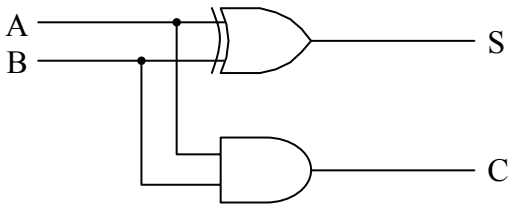



**Fig. 7.1**. Half adder

The operation of subtracting two bits is described similar to the above. Consider finding the difference of two bits, X-Y. Again, there are two results of this operation: the difference bit, D, and the borrow bit, B. The rules of binary subtraction lead to the algebraic description shown in equation (7.2). One possible circuit implementation is shown in Fig. 7.2.

$$D = \overline{X}\,Y + X\,\overline{Y} = X \oplus Y \qquad (7.2)$$
$$B = \overline{X}\,Y$$

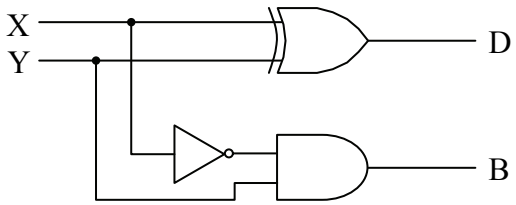



**Fig. 7.2**. Half subtractor

The general problem of designing a circuit to add two N-bit numbers is solved by the same process used for manual addition. Two corresponding bits from the numbers are added together with the carry generated from the addition of the two preceding less significant bits. Of course, no carry is used in the addition of the least significant bits (a half adder could therefore perform this addition). Therefore, the general design solution requires a basic circuit with three binary inputs (two number bits and the carry input) and two outputs, the sum bit and the carry bit. Such a circuit is called a full adder. K-maps for the full adder output functions can be constructed as shown in Fig. 7.3 using the rules of binary addition. In this figure, C represents the carry input while A and B represent corresponding bits from the numbers to be added.

| | $\overline{A}\,\overline{B}$ | $\overline{A}\,B$ | $A\,B$ | $A\,\overline{B}$ |
|---|---|---|---|---|
| $\overline{C}$ | | 1 | | 1 |
| $C$ | 1 | | 1 | |

*(a) S = Sum bit*

| | $\overline{A}\,\overline{B}$ | $\overline{A}\,B$ | $A\,B$ | $A\,\overline{B}$ |
|---|---|---|---|---|
| $\overline{C}$ | | | 1 | |
| $C$ | | 1 | 1 | 1 |

*(b) $C_0$ = Carry out bit*

**Fig. 7.3**. Full adder output functions

The sum and carry out functions are best analyzed using exclusive-OR relationships. The minimized SOP expression is:

$$S = \overline{A}\,B\overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\,BC = \overline{A}\,(B\overline{C} + \overline{B}C) + A\,(\overline{B}\,\overline{C} + BC) =$$
$$= A\,(B \oplus C) + A\,\overline{(B \oplus C)} = A \oplus (B \oplus C)$$

and

$$C_0 = A\,B\,\overline{C} + \overline{A}\,B\,C + A\,B\,C + A\,\overline{B}\,C = A\,B(\overline{C} + C) + C(\overline{A}\,B + A\,\overline{B}) =$$
$$= A\,B + C(A \oplus B)$$

The corresponding full adder circuit is shown in Fig. 7.4.

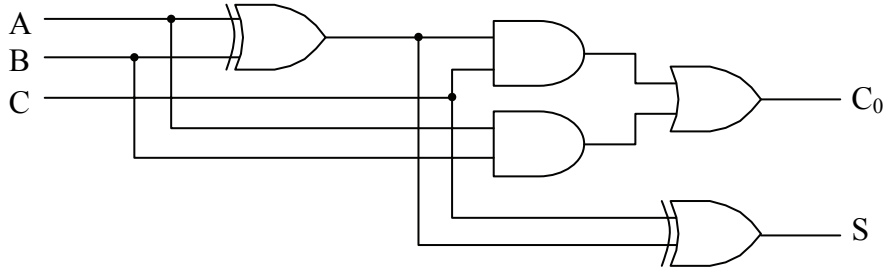



**Fig. 7.4**. Full adder circuit

Using full adder circuits as building blocks, circuits can be designed for the parallel addition of N-bit binary numbers. The most straightforward approach is shown in Fig. 7.5 for a 4-bit adder. Note that a full adder is shown to add the least significant bits, $A_0B_0$. This design would permit 4-bit adders to be cascaded to form larger adding circuits. In this application, the most significant sum bit, $S_4$, would form the carry out to the next more significant 4-bit adder. Of course, regardless of how many bits are added in a given application, the carry input to the circuit adding the least significant bits of the numbers must be zero.

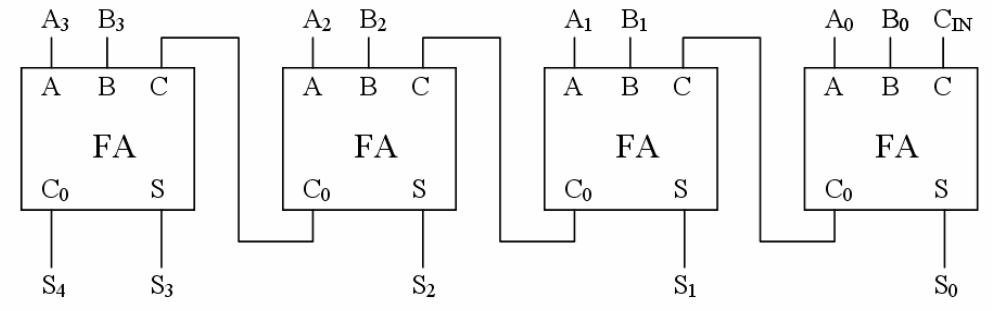



**Fig. 7.5.** 4-bit full adder

The circuit shown in Fig. 7.5 is frequently called a ripple carry adder because the sum does not become valid until the carry propagates or "ripples" through the chain of full adders. This property is the major disadvantage of this circuit since it makes the circuit slow to respond to changes in the inputs. For a long

ripple carry adder this problem can be very severe. From Fig. 7.4 it can be seen that the carry output circuit is not a 2-level circuit. Since the exclusive-OR is a complex gate, it has a longer propagation delay than other basic SSI gates, so that the carry output function of the full adder circuit shown in the figure cannot respond to changes in the inputs until after a delay in excess of three gate propagation delays. In a 16-bit ripple carry adder, for example, the sum would not be valid for a change in inputs until after a delay well in excess of 48 propagation delays. Where fast adder circuits are required, the ripple carry adder is clearly not adequate. In practice, carry look ahead circuits are used which produce the carry bits directly from the inputs rather than waiting for the carry to propagate through the chain. This considerably increases the speed of the adder circuit but it does require additional circuitry.

The above discussion focuses mainly on the design of addition circuitry. A full subtractor circuit could be designed using the same process as was used for the full adder. However, adder circuits can be used to perform subtraction if the number being subtracted (subtrahend) is changed to its two's-complement.

Many of the commonly required arithmetic functions are available as medium scale integrated (MSI) circuits in both the TTL and CMOS families. Whenever possible, these chips should be used in practical designs to minimize chip count.

A typical example is the 7483/74283 adder chip shown in Fig. 7.6 (CMOS equivalents are the 4008 and 74HC283). This chip adds two 4-bit words, A and B, and a carry input, CI, to produce a 4-bit sum, Σ, and a carry out, CO. The carry out function is generated by carry look ahead circuitry so that the maximum specified propagation delay from carry in to carry out is just 20 nsec. The CI and CO connections permit the chip to be cascaded to form longer adders. If the 7483 is used to add only 4-bit words, the CI input must be held low.

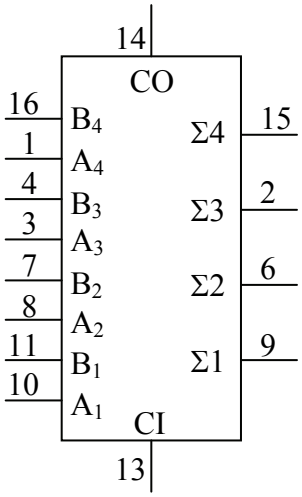



**Fig. 7.6**. 7483/74283 adder

## LABORATORY ASSIGNMENTS

### Assignment 7.1

1) *Design the half adder*. Build a corresponding 2-level circuit using only NAND gates and inverters.

2) *Connect* toggle switches to the two inputs and LED monitors to both outputs. Test the half adder, using Table 7.1 to record your observations on the Sum and Carry outputs.

**Table 7.1**. Test results for the half adder circuit

| Inputs | | Outputs | |
|---|---|---|---|
| A | B | S | C |
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

3) *Design* the *full adder*. Using the K-maps in Fig. 7-3, find minimized SOP expressions for the sum and carry outputs. Build a corresponding 2-level circuit using only NAND gates and inverters.

4) Connect toggle switches to each input and LED monitors to both outputs. Test the full adder, using Table 7.2 to record your observations on the Sum and Carry outputs.

**Table 7.2**. Test results for the full adder circuit

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | $C_i$ | S | C |
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

5) *Two-bit ripple adder*. Connect the half adder and full adder together, according to the diagram shown in Fig. 7.7 to form a two-bit ripple adder.

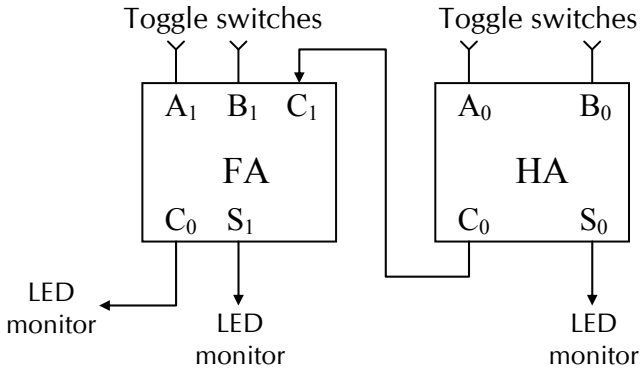



**Fig. 7.7**. 2-bit ripple adder

Test the operation of the adder by setting the toggle switches to several different values and observing the sum and carry indicated by the LED monitors. Demonstrate the circuit operation for your instructor.

6) *7483 IC adder operation*. Install a 7483 IC on the circuit board, and make the following connections:
   a) Connect $V_{CC}$ to + 5 V and GND to power ground.
   b) Connect $C_0$ to power ground.
   c) Connect toggle switches to inputs $A_0$ through $A_3$ and $B_0$ through $B_3$.
   d) Connect LED monitors to sum outputs $S_0$ through $S_3$ and also to $C_4$.

7) Verify that the adder is operating correctly by entering the input values listed in Table 7.3 and recording your observations on the outputs in the table.

**Table 7.3**. Test results for the 7483 IC

| Inputs | | | | | | | | Outputs | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $A_3$ | $A_2$ | $A_1$ | $A_0$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ | $C_4$ | $S_3$ | $S_2$ | $S_1$ | $S_0$ |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | | | | | |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | | | | | |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | | | | | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | |

**Assignment 7.2**

Construct the K-maps for the difference and borrow out functions of a full subtractor. Design, build and test a corresponding circuit using XOR gates wherever appropriate.

**Assignment 7.3**

Design a circuit using the 7483/74283 to add two binary numbers whose values can range from 0 to 63. The adder must be able to add two 6-bit numbers in order to compute values up to 63.

The adder must be able to add two 6-bit numbers in order to compute values up to 63. This requires two 7483/74283 adders cascaded together. In the cascaded operation the unused input bits must be tied to ground for a logic LOW, equivalent to adding a 0. This is necessary to obtain the correct results since unused TTL inputs float to a logic HIGH, which is equivalent to adding a 1.

**Assignment 7.4**

Design an adder/subtractor that can compute values between the range of +63 and –63 and never result in an overflow condition, using the 7483/ 74283, the 7486 and any additional circuitry.

The range of results that must be accommodated so that overflow does not occur is (±63)×2, or +126 to –126. The result requires a total of 8-bits, including the sign bit. The input values will require only 7 bits, including the sign bit.

In the space provided below, draw the corresponding designed circuit and record the test results.

**Assignment No ___.**

**Designed circuit:**

**Test results:**

EXPERIMENT

# 8

# PROPERTIES OF LATCHES/FLIP-FLOPS

## OBJECTIVES

1. Understand the operation and properties of commonly used latch and flip-flop circuits.
2. Conduct laboratory tests to verify the operation of latch and flip-flop circuits.

## EQUIPMENT REQUIRED

Digital Logic Trainer
Dual trace oscilloscope
7400, 7402, 7474, 7475, 7476, 74LS76, 74279 integrated circuits
Other integrated circuits and components, as required

## BASIC INFORMATION

All of the previous experiments have focused on combinational circuits, i.e., circuits in which the state of the output at any given time depends only on the combination of values on the inputs at that time. More generally, digital circuits are sequential in nature. In sequential circuits the state of the output at any given time is determined by the combination of values on the inputs at that time and at previous times. Sequential circuits are therefore characterized by the property of memory.

The basic logic circuit component that provides the property of memory is the flip-flop. The flip-flop is a fundamental digital circuit component that has two stable states. Its output can be either HIGH or LOW depending on the *sequence* of logic levels that has been previously applied to the inputs. There are a variety of common flip-flop devices used in digital circuits.

Figures 8.1 (a) and (c) show two basic flip-flop circuits, called latches, which are designed from gates. First consider the NOR latch in (a). Assume initially that a logic LOW is applied to both the S and R inputs. If a logic HIGH is now

applied to the S input the $\overline{Q}$ output is forced LOW. This level is fed back to the lower gate and the Q output is forced HIGH. Now assume that the S input is allowed to return to a LOW level. Note that nothing changes in the outputs due to the feedback. The $\overline{Q}$ output remains forced LOW and the Q output remains HIGH. This is one of the two stable states of the circuit, usually referred to as the SET state. By similar reasoning, if a high level is now applied to the R input, the Q output is forced LOW and the $\overline{Q}$ output is forced HIGH. If the R input is now returned to a LOW level, again there is no change in the outputs. This is the second stable state, or RESET state, of the circuit. From the above it follows that, if S and R are both LOW, the state of the outputs is determined by which input was last asserted HIGH, i.e., the circuit has "memory". This condition on the inputs is usually referred to as the *hold* or *rest* condition. If both inputs are asserted HIGH, both outputs are forced LOW; this condition is not used in applications. The NAND latch shown in (c) operates in a similar way except the inputs are active LOW. The hold condition occurs when both inputs are HIGH; a LOW on the S input sets the latch (Q = 1); a LOW on the R input resets the circuit. Logic symbols for these latches are shown in (b) and (d).

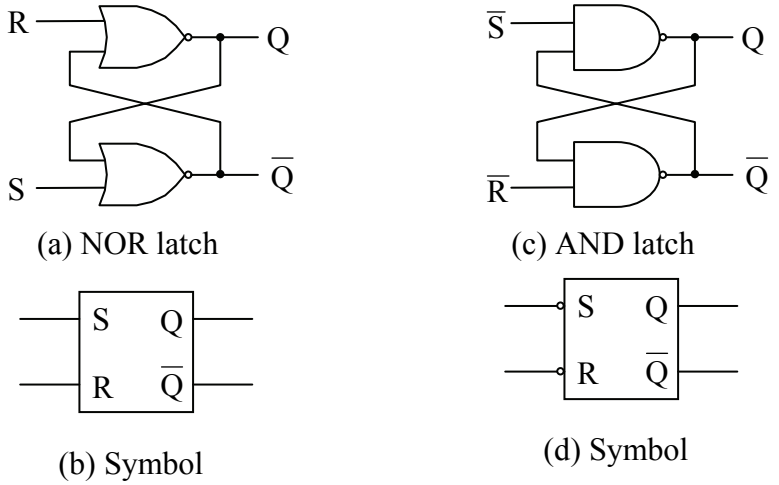



**Fig. 8.1**. Basic latch circuits

A number of latch circuits are available as integrated circuits. For example, the 74279 is a quad NAND latch chip providing four NAND latch circuits. The 7475 is a quad latch in which the latch circuits are "transparent". Fig. 8.2 shows a logic symbol for one of the latch circuits contained in the integrated

circuit. Data are applied to the D input. As long as the E (enable) input is HIGH, the Q output is the same as the D input (in this sense, the latch is transparent, since its input value can be seen from the outputs). When the E input is forced LOW, however, the data value present at the D input is latched; the Q output will remain equal to the D input value at the time the E input was forced LOW.

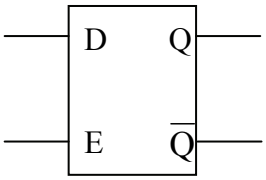



**Fig. 8.2**. 7475 latch

Most integrated circuits in common use have a clock input which controls when the outputs change state in response to the data inputs. There are generally two classifications of sequential circuits, asynchronous and synchronous. In asynchronous circuits outputs change as soon as the logic inputs change, whereas in synchronous circuits outputs can only change in response to a timing, or clock, signal. Asynchronous circuits are frequently subject to operational errors due to the effects of circuit delays and the unpredictable times at which circuit flip-flops change state. In contrast, synchronous circuits can only change state when a clock pulse occurs; their operation is more orderly and their design is more straight-forward.

An example of a clocked D-type flip-flop is the 7474 integrated circuit which contains two flip-flops. The flip-flop logic symbol is shown in Fig. 8,3. This flip-flop has most of the features found on current integrated circuit flip-flops. There are two sets of inputs. The D and CP inputs are synchronous, whereas the $\overline{S}_D$ and $\overline{R}_D$ inputs are asynchronous. If these inputs are inactive, the flip-flop state is determined by CP and D: Q will become equal to the value (1 or 0) on the D input when the clock input (CLK) makes a LOW to HIGH transition. This device is an example of an edge-triggered flip-flop since state changes are synchronized to the leading (LOW to HIGH) edge of the clock pulse. The edge triggered nature of the flip-flop is signified by the triangle on the CP input. Edge triggered devices are also available which are triggered on the trailing (HIGH to LOW) edge of the clock; in this case an inverting bubble would be shown on the clock input line.

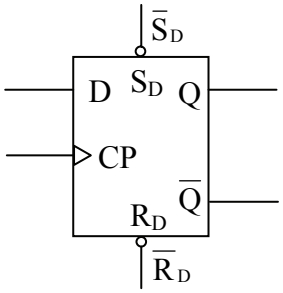



**Fig. 8.3.** 7474 flip-flop

The most versatile flip-flop is the JK flip-flop. The 74LS76 is an integrated circuit containing two JK flip-flops. Fig. 8.4 (a) shows the logic symbol for the flip-flop and (b) shows the function table summarizing its operation. Note that the synchronous J and K inputs are triggered on the trailing clock edge. As with the 7474, there are asynchronous, active low direct set and reset inputs that over-ride the clock and J and K inputs. The main advantage of JK logic compared to SR logic is that all four possible combinations of the levels on the JK lines produce useful operation. It can be seen from the function table that, in addition to hold, store 1 and store 0 operation, the input condition where J = K = 1 produces a toggle mode of operation, i.e., the flip-flop changes state after the active clock edge. It is important to note that the 7476 IC, in contrast to the newer 74LS76, is not actually an edge triggered device but rather a master-slave flip-flop.

## LABORATORY ASSIGNMENTS

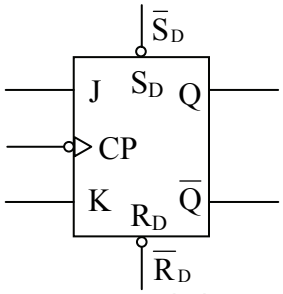
### Assignment 8.1

a) *NOR latch:* Wire the NOR gate latch shown in Fig. 8.1 (a). Connect normally LOW pushbutton switches to the R and S inputs of the circuit. You will monitor circuit outputs Q and $\overline{Q}$ with LED monitors.

b) Turn the power supply on, and note the states of both LEDs:
$Q = \_\_$ ; $\overline{Q} = \_\_$ .

*Predicting the states of a latch when power is first applied is impossible, so the values just recorded are random.*

Clear Q by momentarily pulsing the R input HIGH. If Q is already HIGH, pulsing the R input will have no effect on the circuit.




*(a) Logic symbol*

| Mode description | Inputs | | | | | Output Q |
|---|---|---|---|---|---|---|
| | $\overline{S}_D$ | $\overline{R}_D$ | $\overline{CP}$ | J | K | |
| Asynchronous set | 0 | 1 | × | × | × | 1 |
| Asynchronous reset | 1 | 0 | × | × | × | 0 |
| Hold | 1 | 1 | ↓ | 0 | 0 | Q |
| Store 0 | 1 | 1 | ↓ | 0 | 1 | 0 |
| Store 1 | 1 | 1 | ↓ | 1 | 0 | 1 |
| Toggle | 1 | 1 | ↓ | 1 | 1 | $\overline{Q}$ |

*(b) Function table*

**Fig. 8.4**. 74LS76 flip-flop

c) Pulse the S input HIGH, and observe the effects on the circuit outputs:
Q = __; $\overline{Q}$ = __.

Note that releasing the pushbutton does not cause Q to change from its new state. Why? _____.

Now pulse the S input HIGH again. What effect does this have on the circuit
outputs?_____.

d) Pulse the R input HIGH, and observe that Q changes back to LOW and stays LOW even after the pushbutton is released.

e) Alternatively pulse the S and R inputs HIGH several times. Note that the outputs are always at opposite states.

f) Press and hold the S and R inputs HIGH at the same time. Note that both outputs are now LOW. Release the pushbuttons, and note the states of the outputs. Are they both still LOW? _____.

**Assignment 8.2**

g) 7475 IC D latch operation. Note that the 7475 has four D latches. The latch CLK inputs are tied together in pairs resulting in dual two-bit D latches. You will use only one of the D latches for this experiment, so examine Fig. 8.5 closely for the proper connections to be made.

h) Install a 7475 IC on the circuit board, and make the connections shown in Fig. 8.5. Connect a toggle switch to D1, a normally LOW pushbutton switch to CLK, and LED monitors to Q1 and $\overline{Q}_1$. When the circuit is completed, perform the following steps:

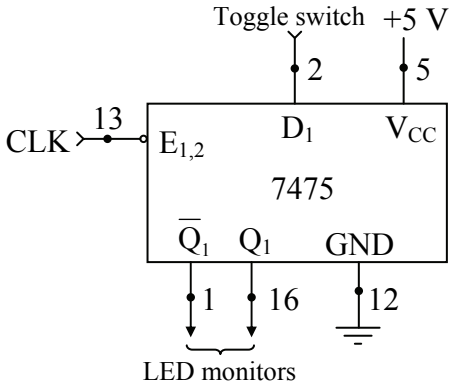



**Fig. 8.5**. The7475 IC D latch

1) Turn the power supply on and monitor the outputs of the latch. Change the toggle switch back and forth a few times, and note that there is no effect on $Q_1$. This is because the latch is in the latch mode, and the data inputs are not enabled. Set $D_1 = 0$.

2) Press and hold the CLK input HIGH. Observe that $Q_1$ is LOW. Change $D_1$ back and forth a few times. What happens to $Q_1$? _____. Now set $D_1 = 1$, and release the CLK pushbutton. What happens to $Q_1$? _____.

3) Change $D_1$ back and forth several times. Observe that $Q_1$ does not change. This proves that the data at $D_1$ is *latched* on the negative-going transition of the clock signal and that the output at $Q_1$ *follows* the data at $D_1$ while the clock signal is HIGH.

**Assignment 8.3**

i) Edge-*triggered D flip-flop – 7474 IC*. The 7474 IC has two individual positive edge-triggered D flip-flops with separate clock inputs and DC SET and DC RESET inputs.

Install a 7474 IC on the circuit board, and make the following connections to one of the D flip-flops:

1) Connect $V_{cc}$ and DC SET to +5 V, GND to power ground.
2) Connect a toggle switch to the D input.
3) Connect a normally HIGH pushbutton switch to the CLK input.
4) Connect a normally HIGH pushbutton switch to DC RESET.
5) Connect LED monitors to Q and $\overline{Q}$ (or monitor the outputs with a logic probe).

j) *7474 synchronous operation:* Apply power and monitor the Q output. Observe that nothing happens when you toggle the D input switch back and forth. This is because the D input is a synchronous input that operates with the CLK input.

Clear Q to 0 by momentarily pulsing the DC RESET input LOW. Set D to 1, and apply a negative-going transition at CLK. Do this by pressing and holding the CLK pushbutton LOW. What happens to Q? _____.

Now apply a positive-going pulse at CLK by releasing the pushbutton switch. What happens? _____. This proves that the flip-flop responds only to positive-going transitions.

Make D = 0, and pulse CLK momentarily. This should clear Q back to 0.

k) *7474 asynchronous operation:* For both DC SET and DC RESET, verify the following:

1) The inputs are active LOW and do not require a pulse at CLK to become activated.
2) The inputs override the synchronous input signals.

**Assignment 8.4**

*Edge-triggered JK flip-flop 74LS76 IC*. Install a 74LS76 IC on the circuit board, and make the following connections:

1) Connect $V_{cc}$ and DC SET to $+5$ V, GND to power ground.
2) Connect toggle switches to J and K inputs.
3) Connect a normally LOW pushbutton switch to the clock input.
4) Connect a normally HIGH pushbutton switch to DC RESET.
5) Connect LED monitors to outputs Q and $\overline{Q}$ (or use a logic probe to monitor the outputs).

Turn the power supply on, and observe the states of Q and $\overline{Q}$. If Q = 1, then pulse DC RESET momentarily LOW. Note that this input clears the flip-flop immediately without a clock signal and that the input is active LOW.

l) *74LS76 synchronous operation:* In this step, you will observe that the J and K inputs can be used to change the output state of the flip-flop. You will also observe that in order for these inputs to effect a change, a clock pulse must be applied. For this reason, the J, K, and CLK inputs are referred to as *synchronous* inputs. Verify this by performing the following steps:

   1) Change the J and K input switch settings, and observe that nothing happens to Q.
   2) Set J = 1 and K = 1, and apply a positive-going transition at CLK. Do this by pressing and holding the CLK pushbutton switch. What happens to Q? _____.
   3) Repeat step 2 using a negative-going transition at CLK. Do this by releasing the pushbutton switch. What happens to Q? _____. This proves that the flip-flop responds to only negative-going transitions. Apply several more pulses to the CLK input. What happens? _____.
   4) If Q is LOW, pulse the CLK input so that Q is HIGH. Set J = K = 0, and note that nothing happens to Q. Pulse the CLK input momentarily, and observe that nothing happens to Q. Why? _____.
   5) Set J = 0 and K = 1, and note that nothing happens to Q. Pulse the CLK input momentarily. What happens to Q?_____.

      Apply several more pulses to the CLK input, and observe the Q remains in the LOW state.

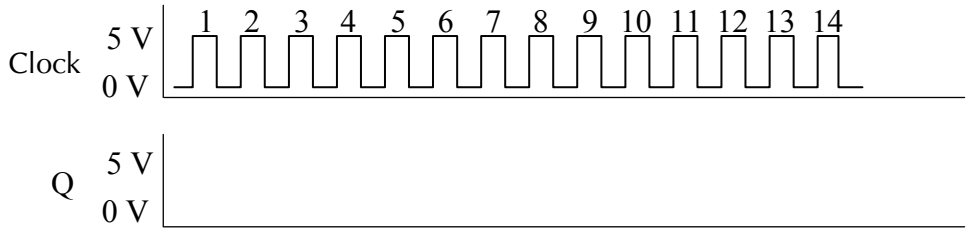
6) Change J to 1 and then back to 0, and note that nothing happens to Q. Pulse the CLK input momentarily. You should observe that Q remains LOW. This proves that the J and K input states present *at the time of the proper clock transition* are the ones transferred to the flip-flop output.

7) Set J = 1, K = 0. Note that nothing happens to Q. Apply a clock pulse, and observe that Q will go HIGH. Apply several more clock pulses. What happens to Q?

_____.

m) Disconnect the pushbutton switch at the CLK input, and replace it with the output of a square wave generator set to 1 MHz (or the highest frequency obtainable). Connect the oscilloscope to observe the clock signal and output Q. Draw the waveforms displayed on the oscilloscope on Timing Diagram 8.1.

Verify that the flip-flop changes states on the negative-going transitions and does not change states on the positive-going transitions. What is the frequency of the Q waveform compared to the clock waveform? _____.




***Timing diagram 8.1***

n) *74LS76 asynchronous operation:* The DC SET and DC RESET inputs are *asynchronous* inputs that operate independently from the synchronous inputs (J, K, and CLK). The asynchronous inputs *override* the synchronous inputs when activated. Verify this by holding the DC RESET input LOW and observe that the flip-flop output stops toggling even though clock pulses are still being applied. Q will remain LOW, until the first clock pulse after the DC RESET pushbutton is released.

o) Disconnect the jumper connection from DC SET to $V_{cc}$ at the $V_{cc}$ end only, and touch this wire to ground. You should now observe that the flip-flop output stops toggling and remains HIGH as long as DC SET is held LOW.

**Assignment 8.5**

p) *7476 master/slave J-k flip-flop operation:* The 7476 IC identical to the 74LS76 IC that was tested in Assignment 4, except that the flip-flop circuits are pulse-triggered instead of edge-triggered. This will permit you to observe the differences between edge-triggered flip-flops and master/slave flip-flops.
Install a 7476 IC on the circuit board, and make the following connections to one of the J-K flip-flops:

1) Connect $V_{cc}$ and DC SET to $+5$ V, GND to power ground.
2) Connect toggle switches to the J and K inputs.
3) Connect a normally LOW pushbutton CLK.
4) Connect a normally HIGH pushbutton switch to DC RESET.
5) Connect LED monitors to Q and $\overline{Q}$ (or monitor the outputs with a logic probe).

q) *7476 synchronous operation:* To test the synchronous operation of the 7476, do the following steps:

1) Set J = 1 and K = 0. Turn the power on and note the states of Q and $\overline{Q}$. If the flip-flop is not cleared (Q = 0), then pulse the DC RESET input LOW momentarily.
2) Press and hold the CLK input HIGH. You should observe that this has no effect on the outputs. Now release the pushbutton. What happens to Q? _____.
Pulse the CLKinput several more times, and note that this has no effect on the outputs.
3) Change J to 0. Note that this has no effect on the outputs. Pulse the CLK input several times. You should observe that this also has no effect on the outputs. Why?_____.
4) Change K to 1, and note that Q does not change. Press and hold the CLK input HIGH. What happens to Q? _____. Now release the CLK pushbutton. What happens to Q now?_____.
Pulse the CLK input several more times, and note that Q does not change.
5) Change J to 1. Note that Q remains LOW. Press and hold the CLK pushbutton HIGH. What happens to Q? _____.
Release the pushbutton. What happens to Q now _____. Pulse the CLK input several more times. You should observe that Q changes states on each CLK pulse.

r) In step q, you should observed that the flip-flop loaded the J and K inputs only when the CLK is high, and they were transferred to Q and $\overline{Q}$ on a negative-going transition at CLK. Now you will observe the chief disadvantage of the master/slave: data at the J and K inputs can affect the flip-flop outputs any time while the CLK input is HIGH.

SET J = 0 and K = 1. Clear the flip-flop by momentarily pulsing DC RESET to LOW. Press and hold the CLK pushbutton HIGH. Change J to 1 and then back to 0. Noe release the pushbutton. You should observe that Q changes to 1 even though J = 0 and K = 1 at the time of the negative-going transition. This demonstrates that, should an unwanted glitch or noise spike occur on J or K while the CLK input is HIGH, it may cause the flip-flop outputs to be invalid when CLK goes LOW.

EXPERIMENT

# 9

# RIPPLE COUNTER DESIGN

## OBJECTIVES

1. Design and understand the basic operation of binary ripple counters. Design ripple counters of any modulus.
2. Investigate the application of J-K flip-flops on counting circuits.
3. Investigate the operation and a method of changing the mod-number of the 7493 IC counter.

## EQUIPMENT REQUIRED

Digital Logic Trainer

Dual trace oscilloscope

Integrated circuits and components, as required

## BASIC INFORMATION

A *counter* is a circuit consisting of a number of flip-flops and gates working together to count the number of clock pulses applied to its input. Counters are available in two categories: ripple (asynchronous) counters and synchronous counters. In a ripple counter the flip-flop output transition serves as a source for triggering other flip-flops. Clock input is applied to only the first of the series of flip-flops. Clock pulses for the other flip-flops come from the preceding flip-flop. Thus, the clock pulse "ripples" through the circuit in series fashion. In a synchronous (parallel) counter, the clock inputs of all of the flip-flops receive the common clock pulse, and the change of state is determined from the present state of the counter. Although ripple counters are subject to timing constraints and glitch problems, they are often easier to design and require less hardware than corresponding synchronous circuits (to be considered in Experiment 10). As long as designers understand the problems and limitations of ripple counter circuits, and use them appropriately, ripple counter circuits offer simple and economical solutions to many practical design problems.

A basic 3-bit ripple counter circuit is shown in Fig. 9.1. $\overline{CLK}$ is the circuit input and the outputs are labeled ABC. Note that the JK flip-flops are set to toggle mode. Flip-flop C will toggle for every pulse arriving on the input. However flip-flop B can toggle only when C makes a HIGH to LOW transition, and the same characteristic is true for flip-flop A relative to B. Fig. 9.2 shows the waveforms that would be observed at the outputs assuming that the input is a symmetrical square wave.

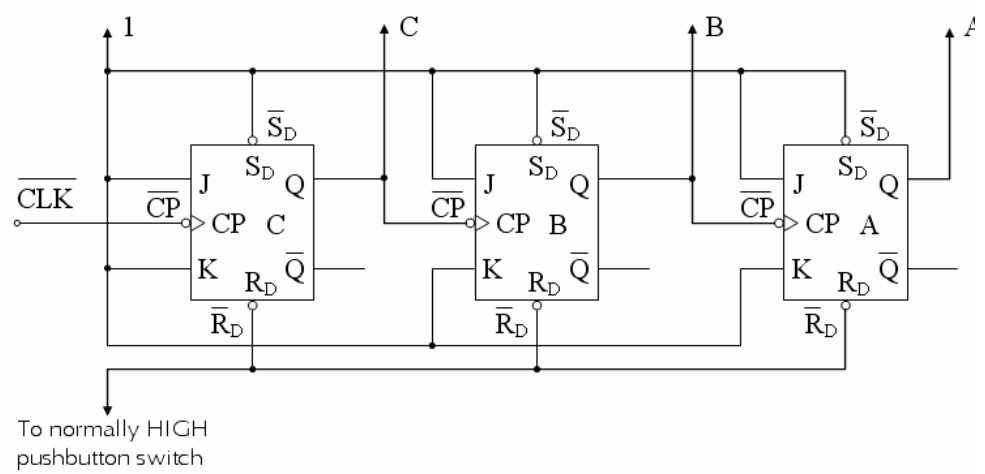



**Fig. 9.1**. 3-bit ripple counter

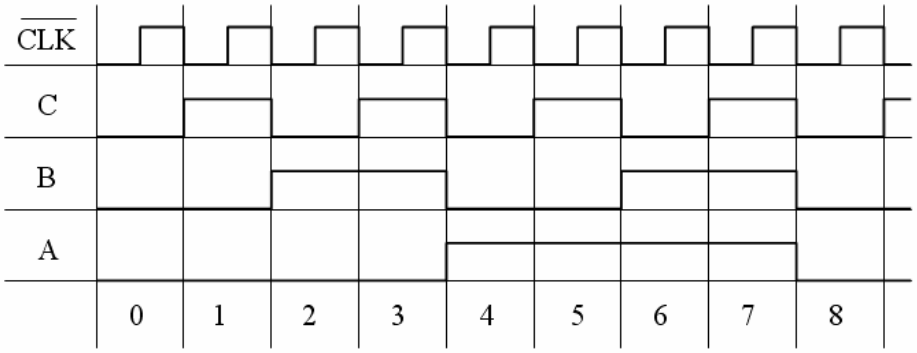



**Fig. 9.2**. 3-bit ripple counter waveforms

A number of characteristics of this basic circuit are important to note. The counter counts up in binary from 000 to 111 and then recycles. The number of states of the circuit, referred to as the *modulus* or *mod-number* of the counter, is 8, corresponding to $2^3$. The Q output of each flip-flop is a square wave with

a frequency output of one half that of the input signal. The frequency output from the final stage is $1/2^3$. These characteristics can be generalized. For a binary ripple counter with N flip-flops, the modulus is $2^N$, the terminal count is $(2^N-1)$, and the final output frequency is the input frequency divided by $2^N$.

The number of output bits of a counter is equal to the flip-flop stages of the counter. A MOD-$2^N$ counter requires N stages of flip-flops in order to produce a count sequence of the desired length. The first stage of a counter is the *least significant bit* (LSB). The last stage of a counter is the *most significant bit* (MSB).

There are many applications where a down-counter is required, i.e., a counter which counts downward in binary to a count of 0 before recycling to its initial count. Note that in the above circuit down-counter operation would be obtained if the counter outputs were taken from the $\overline{Q}$ outputs of the flip-flops. Alternatively, the counter outputs could be taken from the Q outputs of the flip-flops but each flip-flop could be clocked from the $\overline{Q}$ output of the preceding flip-flop. This latter scheme is particularly useful in designing a dual mode up/down counter in which the count direction can be changed by a control signal.

The advantage of ripple counters is their simple hardware. But they are asynchronous circuits and, with added logic, can be unreliable and delay dependent. This is particularly true for logic that provides feedback paths from counter outputs to counter inputs. Also, due to the length of time required for the ripple to occur, large ripple counters are slow circuits. As a consequence, synchronous binary counters are favored in all but low-power designs where ripple counters have an advantage.

Counter applications often require decoding the output count states produced by a counter. Counter decoding can be used to shorten a count sequence, to enable other logic circuits when a specific count state is reached, or to display the count state as a decimal number. The basic binary ripple counter is restricted to a counter modulus which is a power of 2. Many applications, however, require other moduli. There is a simple design solution which permits the design of ripple counters of any modulus. If modulus N is required, the terminal count will be (N-1), after which the counter should recycle to state 0. This can be realized by decoding state N and applying the output of the decoding gate, asserted active low, to the asynchronous reset inputs of the flip-flops. This technique forces the counter back into state 0. Fig. 9.3 shows a mod-6 ripple counter designed in this way. Note that state 6 is

decoded with the NAND gate by using just the two most significant bits of the counter since this is the only state in which both bits would be high.

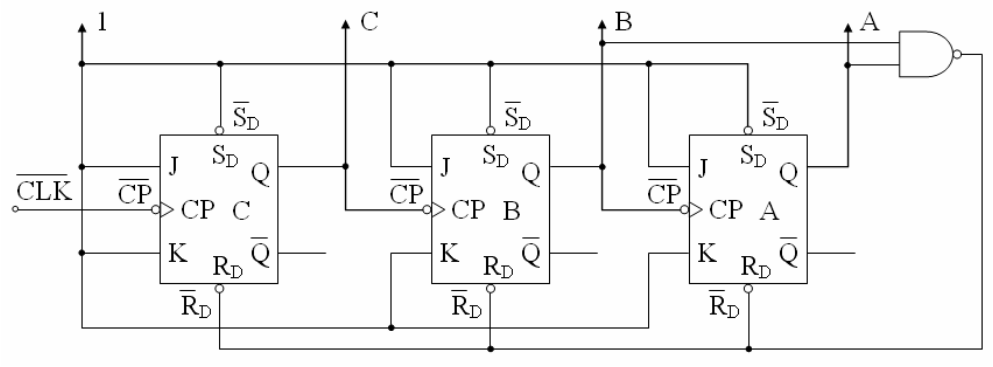



**Fig. 9.3**. Mod-6 ripple counter

## LABORATORY ASSIGNMENTS

### Assignment 9.1

a) Fig. 9.1 shows the circuit for a three-bit binary counter. Examine the circuit closely, then construct it. Arrange the order of the flip-flops exactly as the diagram shows it.

b) Connect a normally HIGH pushbutton switch to the clock input of flip-flop C. Connect LED monitors to the Q outputs of each flip-flop. The order of the LEDs is important, since the output of the leftmost flip-flop will represent the LSB of the count and the rightmost the MSB. Connect all DC SET inputs to $V_{cc}$ and all DC RESET inputs to a single normally HIGH pushbutton switch.

c) Turn the power on, and clear the counter by pulsing the DC RESETs LOW momentarily. The number stored in the counter is indicated by the LEDs, which should all be OFF (i.e., the number should be $000_2$). Test the counter circuit by pulsing the clock input and observing the count indicated by the LED monitors. Record your observations in Table 9.1.

Your results should indicate that the counter counts to a maximum of 7 and recycles to 000 on the eighth clock pulse.

Disconnect the pushbutton switch from the clock input of the counter. Connect the output of a square wave generator, set at 10 kHz, to the clock

input of the counter and to one vertical input of a dual trace oscilloscope. Connect the other vertical input of the oscilloscope to the Q output of flip-flop C. Trigger on output Q of flip-flop C. What is the frequency of the signal at C? _____. Move the oscilloscope input from C to A. Trigger on B. What is the frequency of this signal? _____. Finally, move the oscilloscope input from B to A. Trigger on A. What is the frequency of this signal? _____. Based on your observations, what is the mod-number of this counter? _____.

**Table 9.1**. Test results for the 3-bit ripple counter

| Clock pulse | Output state | | |
|---|---|---|---|
| | A | B | C |
| 0 | 0 | 0 | 0 |
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |

**Assignment 9.2**

*Changing the mod-number of a counter:* Modify the counter circuit so that the wiring is like that shown in Fig. 9.4.

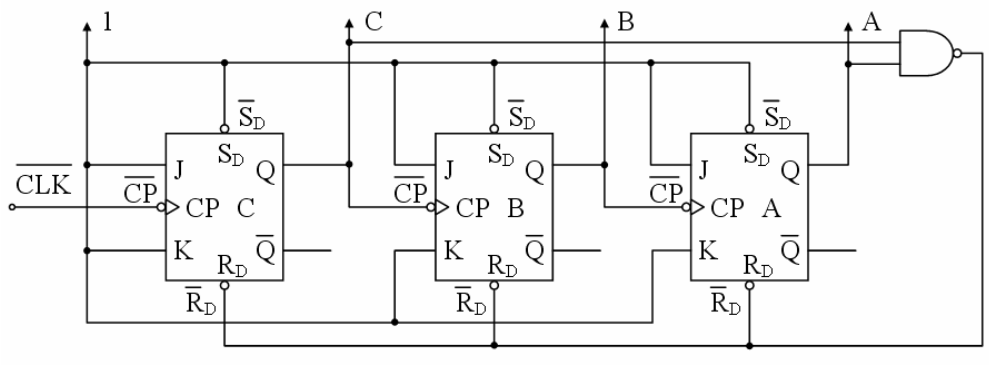



**Fig. 9.4.** Circuit for Assignment 9.2

1) Connect the pushbutton to the counter clock input. Connect LED monitors to the Q outputs of each flip-flop.
2) Clear the counter by pulsing it until the LED monitors indicate a count of 000, or by lifting the DC RESET line connection from the NAND gate output and grounding it momentarily, then reconnecting it to the NAND gate.
3) Using Table 9.2, record the counter output states you observe as you pulse the counter through its new count sequence. Determine the mod-number of this counter by examining the counter sequence: _____.

**Table 9.2.** Determining the mod-number of the counter

| Clock pulse | Output state | | |
|---|---|---|---|
| | A | B | C |
| 0 | 0 | 0 | 0 |
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |

**Assignment 9.3**

a) Refer to the data sheet for the 7493 IC. This IC contains four flip-flops that may be arranged as a mod-16 ripple counter. To do this, $Q_0$ must be tied externally to $\overline{CP}_1$. The MSB of this counter is $Q_3$ and the LSB is $Q_0$. The counter's mod-number may be changed by making the appropriate external connections.
b) *7493 IC operation:* Connect the circuit of Fig. 9.5. Connect a normally HIGH pushbutton switch to input $\overline{CP}_0$ and LED monitors to outputs $Q_3$ through $\overline{Q}_0$.
c) Pulse $\overline{CP}_0$ and observe the counter sequence displayed on the LEDs. It should be count from 0000 to 1111 and then recycle to 0000. Note that the NAND gate inputs $\overline{MR}_1$ and $\overline{MR}_2$ have no effect on the counter, since they are both tied LOW.

d) Disconnect the pushbutton switch at input $\overline{CP}_0$. Connect a square wave generator to this input, and set the generator to 10 kHz. Monitor the $\overline{Q}_3$ output with one vertical input of the oscilloscope and the generator output with the other input. Set the horizontal sweep so that you can verify that there is one $\overline{Q}_3$ pulse for every 16 generator pulses. Is the signal at $\overline{Q}_3$ a square wave? _____.

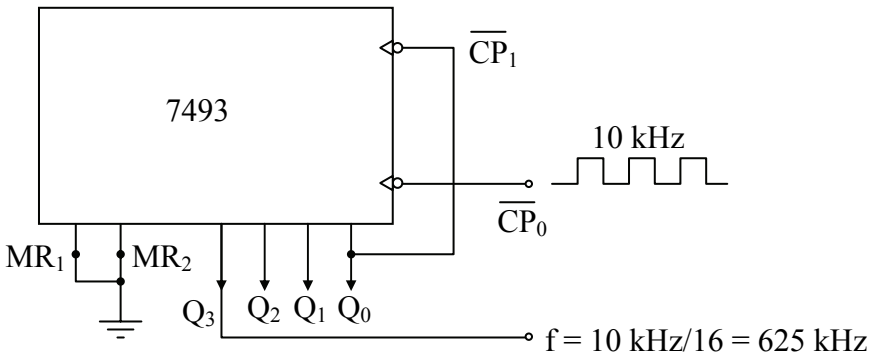



**Fig. 9.5.** The 7493 IC counter

*Changing the 7493 mod-number:* Disconnect the pulse generator from the counter, and reconnect the pushbutton switch in its place. Disconnect $MR_1$ and $MR_2$ from ground and connect one of them to $Q_3$ and the other to $Q_2$.

e) Pulse $\overline{CP}_0$ repeatedly, and observe the count sequence displayed on the LEDs. Record this sequence of output states in Table 9.3.

**Table 9.3.** Observing the count sequence

| Input pulse applied | Output states | | | | Decimal number |
|---|---|---|---|---|---|
| | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | |
| None | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 1 | |
| 2 | 0 | 0 | 1 | 0 | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |
| 15 | | | | | |

f) What is the mod-number of the counter in step j? _____. Verify the mod-number you gave by disconnecting the pushbutton switch at $\overline{CP}_0$ and applying a 10 kHz square wave to this input. Then measure the frequency of the signal at Q3. Is this output a square wave? ____.

g) Now connect the 7493 IC as shown in Fig. 9.6. Repeat steps g-j, using Table 9.4 to record your observations.

h) Based on the results recorded in Table 9.3, determine whether or not the signal at Q3 is a square wave: _____.

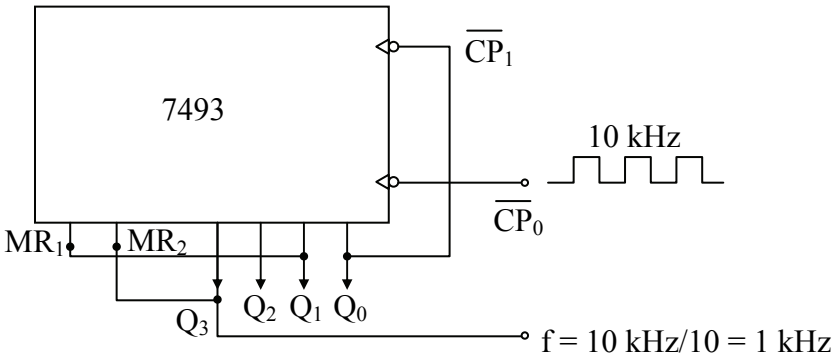



**Fig. 9.6**. The 7493 IC counter

**Table 9.4**. Yest results for the 7493 IC counter

| Input pulse applied | Output states | | | | Decimal number |
|---|---|---|---|---|---|
| | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | |
| None | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 1 | |
| 2 | 0 | 0 | 1 | 0 | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |
| 15 | | | | | |

*i) Cascading 7493 IC counters:* Connect the circuit shown in Fig. 9.7. Apply a 6 kHz square wave to input $\overline{CP}_0$ of the mod-10 counter. With the oscilloscope, determine the frequency of the signal at $Q_3$ of the mod-6 counter: _____. What is the mod-number of this counter arrangement? ____.

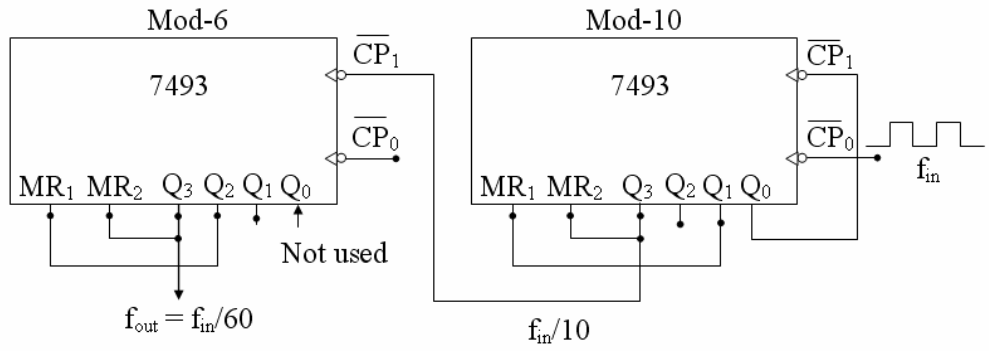



**Fig. 9.6**. Cascading 7493 IC counters

EXPERIMENT

# 10

# SYNCHRONOUS COUNTER ANALYSIS

## OBJECTIVES

1. Completely analyze synchronous counter circuits and represent their operation using a state diagram.
2. Investigate the operation of the 74193 IC counter.

## EQUIPMENT REQUIRED

Digital Logic Trainer

2, 74LS76 and 1, 74193 integrated circuits

Other integrated circuits and components, as required

## BASIC INFORMATION

In synchronous sequential circuits all flip-flops are clocked from the same signal source. In theory, state changes all occur at the same time and the glitch problems common to asynchronous circuits due to propagation delay effects are eliminated. For this reason synchronous circuits are almost always preferable to asynchronous circuits.

The analysis of a synchronous circuit can be conducted in a simple and orderly manner as follows. If there are N flip-flops in the circuit there are a total of $2^N$ states. For each state, following the logic of the circuitry, the logic levels on the J and K inputs for each flip-flop can be determined. Once these are known the next state can be determined. This analysis proceeds until all possible states have been analyzed. It is best to conduct this analysis in tabular form using a table to list the state transitions. The final results of the analysis can best be presented by a state diagram.

The design of natural binary synchronous counters, in which the modulus is a power of 2, is very routine. As binary is counted upward, the least significant bit (LSB) toggles at every change. The next least significant bit toggles when

the LSB is 1. The third least significant bit toggles when both the preceding bits are 1. This pattern continues for counts of any length. Therefore, to design an N-bit binary counter, the design procedure is as follows for J-K flip-flops:

1. Tie the J and K inputs of the LSB flip-flop high so that it always toggles.
2. Tie the J and K inputs of the second least significant flip-flop to the Q output of the LSB flip-flop so that it toggles when the LSB flip-flop is 1.
3. Tie the J and K inputs of any other flip-flop to the AND of the Q outputs of all preceding flip-flops, so that it toggles only when all of them are 1.

The circuit diagram of a 4-bit binary synchronous counter designed with this procedure is shown in Fig. 10.1. The most significant bit (MSB) and LSB flip-flops have been labelled in the Figure. Note that the 3-input AND gate could be replaced by a 2-input gate if the output from the preceding gate were used as input. This procedure could be used for succeeding stages also in longer counters so that only 2-input AND gates would be required. The disadvantage of this practice is that the gate propagation delays accumulate and limit the maximum operating speed of the circuit.

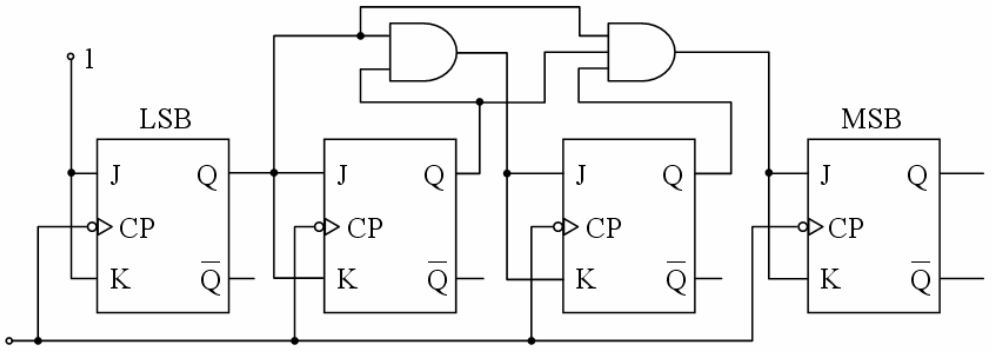



**Fig. 10.1.** 4-bit binary synchronous counter

Synchronous counter design follows a systematic procedure to specify the count sequence required and to determine the input logic functions to obtain the desired count sequence. The flip-flop excitation table describes the input conditions that produce the output state from each individual J-K flip-flop. The excitation table for J-K flip-flops is shown in Table 10.1.

With synchronous counters, *the next output state from each flip-flop is determined by the present state and the present inputs* applied to that stage. The J-K excitation table forms the basis of synchronous counter design. An

excitation table for the counter is constructed by specifying the present state and next state for each flip-flop in the order of the count sequence. For each transition, the necessary J and K inputs to produce the required sequence are listed. A logic function for each J and K input is then derived from the excitation table.

**Table 10.1**. The excitation table for J-K flip-flops

| Present state $Q_N$ | Next state $Q_{N+1}$ | Inputs J | K | Condition |
|---|---|---|---|---|
| 0 | 0 | 0 | x | No change, reset |
| 0 | 1 | 1 | x | Toggle, set |
| 1 | 0 | x | 1 | Toggle, reset |
| 1 | 1 | x | 0 | No change, set |

Synchronous counters can produce a count sequence that is not in counting order. Digital security locks might need a counter that generates a random sequence of numbers to serve as a password that must be matched by an appropriate input. Other applications that can use a nonserial count feature are ones where the counter output is decoded to enable certain circuits within a digital system.

Integrated circuit counters are available for counter applications that require serial up or down count sequences. The IC counters available are 4-bit binary or decade counters that can be cascaded together for applications requiring more than four output states. The IC counters have asynchronous inputs that allow the flexibility of presetting the counter to an initial start value other than zero or resetting the counter back to zero at any instant of time. The 74193 is a 4-bit synchronous, positive edge-triggered, binary counter capable of counting up or down (two separate clock inputs are used to control up or down counting). The counter has a maximum mod-16 count sequence that can be shortened to any modulus less than 16 by using the asynchronous control inputs. The *clear input* resets all count stages back to zero. The *preset input* sets the counter stages to any 4-bit binary number loaded on the *parallel inputs* of the 74193. The counter can be easily cascaded while counting up or down through the *carry output* and *borrow output* to lengthen the modulus and to provide additional counter output stages.

Selection of a counter circuit for a particular application should be based on matching the requirements of the application to the capabilities of the counter. Key parameters that should be checked include counter modulus, maximum clock frequency, clock pulse requirements, cascade inputs and outputs, count enable inputs, and asynchronous preset and clear inputs.

## LABORATORY ASSIGNMENTS

### Assignment 10.1

Theoretically analyze the circuit shown in Fig. 10.2 using a state table technique to determine the circuit operation. Build and test the circuit to confirm your analysis. Connect each of the asynchronous inputs of the flip-flops to a static logic level source so that the circuit can be set to any desired state. Use a push-button manual pulser to supply clock pulses. Use LED indicating circuits on the $\overline{Q}$ outputs to indicate counter state. What is the count sequence? ____. Theoretically and experimentally determine circuit behaviour for all possible states.

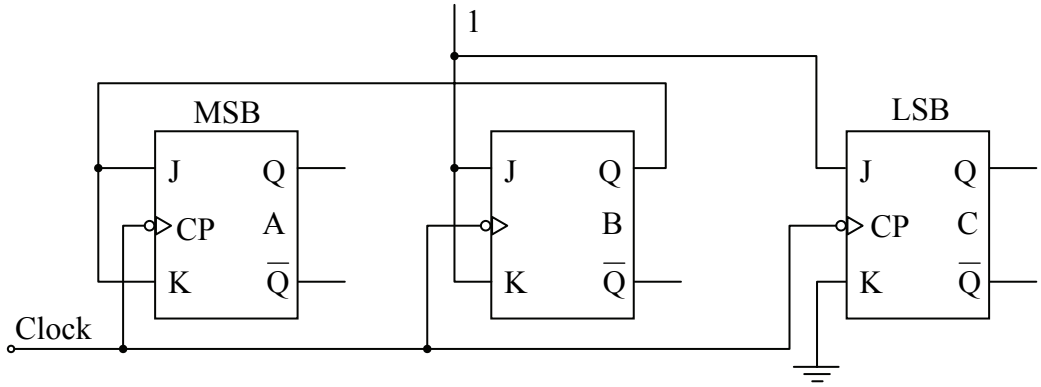



**Fig. 10.2**. Counter circuit for Assignment 10.1

### Assignment 10.2

Design, build and test a glitch-free synchronous mod-6 counter that counts from 000 to 101 and then recycles to 000. Insure in the design that all unused states transition to state 000. Test the circuit as described in Assignment 10.1.

### Assignment 10.3

a) *74193 IC operation as a mod-16 UP counter:* Construct the circuit of Fig. 10.3. Make the following connections to the 74193 IC:
   1) Connect toggle switches to $P_3$ through $P_0$.
   2) Connect normally LOW pushbutton switches to $CP_U$ and MR inputs. (NOTE: if necessary, you may use a toggle switch for MR)
   3) Connect LED monitors to $Q_3$ through $Q_0$ and also at $\overline{TC}_U$.
   4) Connect a toggle switch to $CP_D$.

5) Connect a normally HIGH pushbutton switch to $\overline{PL}$.

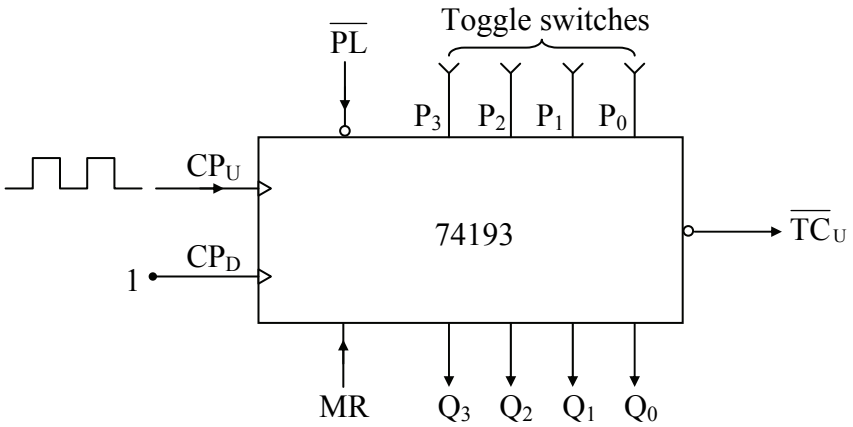



**Fig. 10.3**. Circuit for Assignment 10.3

b) Set the toggle switches to the parallel inputs so that $P_0 = P_1 = P_2 = P_3 = 0$. Set $CP_D$ HIGH. Clear the counter to 0000 by pulsing MR HIGH. Note that $\overline{TC}_U$ (terminal count-up mode) is HIGH.

c) Pulse $CP_U$ HIGH a couple of times, and note that the counter counts UP. Pulse the counter until a count of 1111 is displayed. Record the state of $\overline{TC}_U$: ____. Now pulse the counter one more time. Now record the value of $\overline{TC}_U$: ____. Record the observations you have made in Table 10.2.

**Table 10.2.** Test results for the 74193 IC UP counter

| Input pulse applied | Output states | | | | Decimal number | $\overline{TC}_U$ |
|---|---|---|---|---|---|---|
| | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | | |
| None | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |
| 12 | | | | | | |
| 13 | | | | | | |
| 14 | | | | | | |
| 15 | | | | | | |
| 16 | | | | | | |

d) Set CPD LOW. Pulse the CPU input several times. What happens? _____. Return CPD to HIGH and pulse CPU a few more times. You should observe that the counter does not count as long as CPD is LOW.

e) Using the parallel inputs to preset the counter, set the toggle switches at the parallel inputs so the P3 = P1 = 1 and P2 = P0 = 0 and pulse $\overline{PL}$ LOW. The LEDs should now indicate 1010. Pulse the counter until the $\overline{TC}_U$ output LED indicates 0, observing the output LEDs as you do so. What sequence of numbers does the counter count?

f) _____.
Pulse counter one more time. What is the count now? _____.

g) 74193 IC operation as a mod-16 DOWN counter: Disconnect the toggle switch from CPD and exchange it for the pushbutton switch at CPU and vice versa. Disconnect the LED from $\overline{TC}_U$ and reconnect it at $\overline{TC}_D$. Clear the counter by pulsing MR HIGH momentarily. Pulse the counter through its count sequence, and record your observations in Table 10.3.

**Table 10.3.** Test results for the 74193 IC DOWN counter

| Input pulse applied | Output states | | | | Decimal number | $\overline{TC}_D$ |
|---|---|---|---|---|---|---|
| | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | | |
| None | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |
| 12 | | | | | | |
| 13 | | | | | | |
| 14 | | | | | | |
| 15 | | | | | | |
| 16 | | | | | | |

h) Set the parallel input toggle switches to 1010 and pulse $\overline{PL}$ LOW. Note that $\overline{TC}_D$ is now at 1. Pulse the counter until $\overline{TC}_D$ indicates 0, observing the output LEDs as you do so. What sequence of numbers does the counter count? _____. Pulse the counter one more time. What is the count now? _____.

EXPERIMENT

# 11

# REGISTERS

## OBJECTIVES

1. Understand the general operation of register circuits.
2. Design register circuits using D and J-K flip-flops.
3. Understand the operation and properties of integrated circuit registers.

## EQUIPMENT REQUIRED

Digital Logic Trainer
2, 7474; 2, 74LS76 and 2, 74194 integrated circuits
Other integrated circuits and components, as required

## BASIC INFORMATION

A register is a linear array of flip-flops that is used for storage and simple processing of data. The general structure of hypothetical 5-bit register is shown in Fig. 11.1.

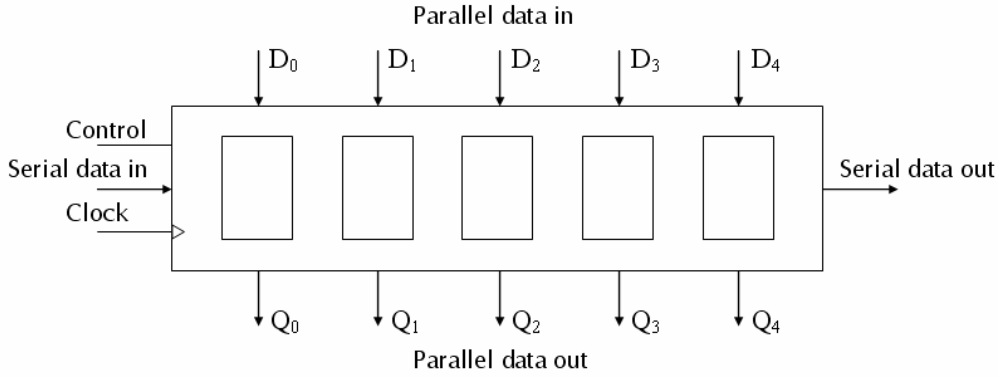



**Fig. 11.1**. General register structure

The register structure outlined in Fig. 11.1 illustrates all of the capabilities that can be found in a register circuit. In this example there are five flip-flops so that the register can store five bits of data which can be input to the flip-flops

in parallel fashion using the inputs $D_0$ to $D_4$, or one bit at a time in serial fashion using the serial data input. Stored data are available either serially, at serial data out, or in parallel at the $Q_0$ to $Q_4$ outputs. Serial operation of the register is accomplished by shifting the data to the right with the bit on the serial data input being stored in the least significant flip-flop, $Q_0$, and the Q output of the most significant flip-flop, $Q_4$, being available at the serial data output. Shift right operation is synchronous while parallel load operation can be either synchronous or asynchronous depending on the register design. Some registers are designed to shift left as well as right; these are known as bidirectional shift registers. In any register circuit with multiple capabilities, the action performed at any time is determined by control inputs.

Registers can be designed using any of the common flip-flop types. Fig. 11.2 (a) shows a simple shift-right register circuit using D flip-flops with serial data input and both serial and parallel data output capabilities. Fig. 11.2 (b) shows the circuit modifications that could be made for each flip-flop to incorporate an asynchronous parallel load capability. Note that in this circuit the asynchronous parallel load control line would be active HIGH and override the clock.

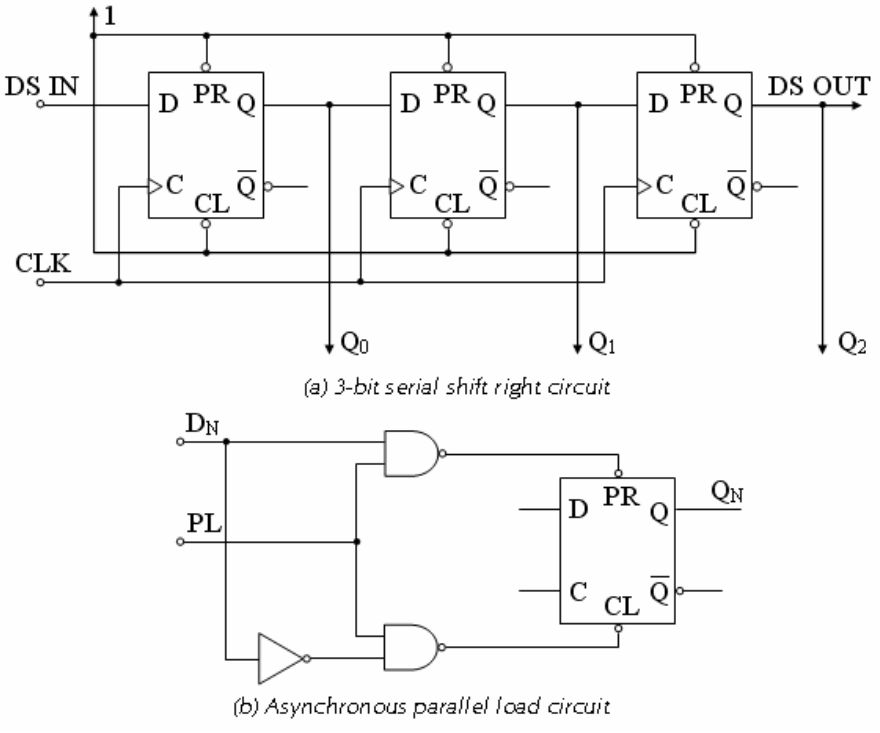



**Fig. 11.2.** Register circuit design with D flip-flops

A wide variety of registers are available as integrated circuits. Fig. 11.3 shows the logic symbols for two 8-bit integrated circuit registers that are especially useful for serial-parallel and parallel-serial conversions. The 74164 features serial data input and parallel data output. Serial data input and shift right operation is synchronized to the leading edge of the clock, CP. Serial data input is gated through the two inputs, DSA and DSB. These may be tied together or one may be used as an active HIGH enable for the other. The active LOW master reset input, MR, clears the register asynchronously and overrides all other inputs when active. The 74165 device features serial and parallel data input and serial data output through $Q_7$. When the parallel load control input, PL, is low, data are loaded into the register asynchronously through the D inputs. When the PL input is high, data enter the register serially through the DS input and shifts one bit to the right synchronously with the positive-going clock transition. The CE input is an active low clock enable; clocking is inhibited when HIGH.

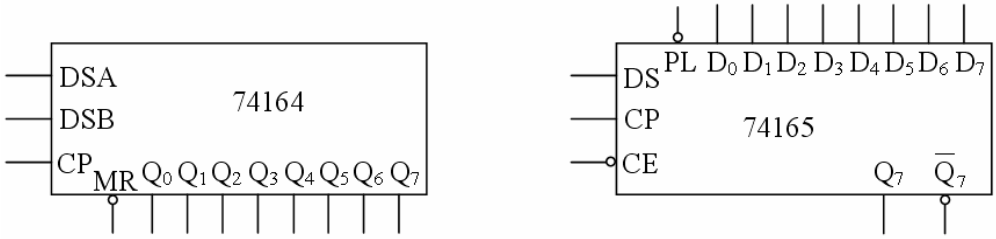



**Fig. 11.3**. 8-bit integrated circuit registers

Fig. 11.3 shows two examples of 4-bit register circuits, which are more versatile than the above devices. The 7495 shown in (a) has parallel and serial input capability as well as parallel and serial output capability. Separate clocks are used for serial input/shift right operation (CP0) and parallel data input (CP1). The operating mode is determined by the level on the select line, S; when S is HIGH data are parallel loaded synchronously with the trailing edge of CP1 (CP0 is a don't care) from the D inputs, but when S is LOW shift right operation occurs with data input serially from the DS input synchronous with the trailing edge of CPA. Even more versatile operation is available with the 74194 circuit shown in Fig. 11.4 (b). This circuit is a bidirectional universal shift register with four distinct operating modes determined by the logic levels on control inputs $S_1$ and $S_0$. When both inputs are HIGH parallel load operation occurs. When $S_0$ is HIGH with $S_1$ LOW, the register shifts right assuming serial data input from $D_{SR}$. For the opposite condition with $S_1$ HIGH and $S_0$ LOW, the register shifts left with serial data input from $D_{SL}$ into the $Q_3$

flip-flop. Finally, when both control lines are LOW the register is placed in a hold or do nothing state; it maintains the stored data regardless of the activities of the clock. Both serial and parallel operations are synchronized to the leading edge of the clock. The active LOW master reset is asynchronous and overrides all other inputs.

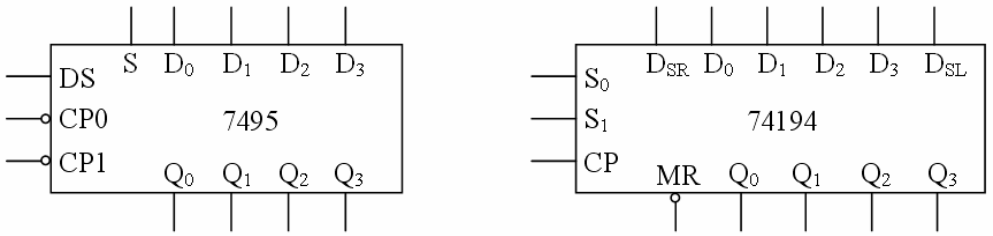



**Fig. 11.4.** Versatile 4-bit registers

The integrated circuit registers discussed above are just four examples of the variety of capabilities available in integrated circuit packages. Although their use simplifies the design of register applications, the designer must be aware that there are generally restrictions on the timing of input transitions on the control lines. For example, HIGH-to-LOW transitions on the control lines, S0 and S1, of the 74194 should only take place when the clock is at a HIGH level. Violating the specified conditions will generally result in unpredictable operation and loss of data.

## LABORATORY ASSIGNMENTS

### Assignment 11.1

Design a serial input shift right register circuit using 74LS76 flip-flops. The circuit must have an asynchronous master reset input which will clear the register when made LOW. Use the pushbutton manual pulser to clock the circuit and static logic level switches to drive the serial data input and the master reset input. Build the circuit and test it to verify correct operation by clocking in a 4-bit combination of serial data and verifying that the correct data have been stored in the register. Test at least 8 different input combinations including the 0000 and 1111 combinations. Remember to test the master reset operation.

**Assignment 11.2**

Design, build and test a 4-bit shift right register with a *synchronous* parallel load capability using 7474 flip-flops. The parallel load must be synchronized to the leading edge of the clock. The operating mode is to be determined by a control input MODE which, when HIGH, will provide parallel load operation but when LOW, will provide serial operation. Adequately test the circuit to provide reasonable assurance of correct operation in both the serial input and parallel load modes.

*Hint:* Base the design on the circuit shown in Fig. 11.2 (b). Synchronous operation can be obtained by also using the clock as input to the NAND gates driving the preset and clear flip-flop inputs.

**Assignment 11.3**

*74194A serial operation*

1) Install a 74194A IC on the circuit board, and make the following connections:
   a) Connect toggle switches to A through D, SR SER, $S_0$, and to $S_1$.
   b) Connect a normally HIGH pushbutton switch to CLK and $\overline{\text{CLR}}$.
   c) Connect LED monitors to $Q_A$ through $Q_D$.
2) Set $S_1$ and $S_0$ to HIGH, SR SER to LOW, and A through D to LOW. Note that this has no effect on the outputs. Pulse $\overline{\text{CLR}}$ LOW momentarily to clear the register if it is not already cleared. Return $S_0$ and $S_1$ to LOW.
3) Verify that neither the serial data input (SR SER) nor the parallel data inputs (A through D) have an effect on the register as long as both $S_0$ and $S_1$ are LOW when CLK is pulsed LOW. Do this by setting SR SER = 1, D = A = 1, and C = B = 0 and momentarily pulsing CLK LOW.
4) Now set $S_0$ to HIGH, and pulse CLK. You should observe that the LEDs indicate 1000. Set SR SER to LOW, and pulse CLK three more times. You should observe that the 1 is shifted one position to the right on each pulse. The output now reads _____.
5) Set $S_1$ HIGH, and pulse CLK LOW momentarily. Observe that the register output changes to the value represented by the parallel data input switches (1001). This type of data transfer is called _____.

*74194A wired as a ring counter*

6) Fig. 11.5 shows the 74194A wired as a ring counter. Examine the circuit, then make the necessary changes to the 74194A so that it is the same as that shown in the figure. Verify that the circuit operates as a four-bit ring counter.

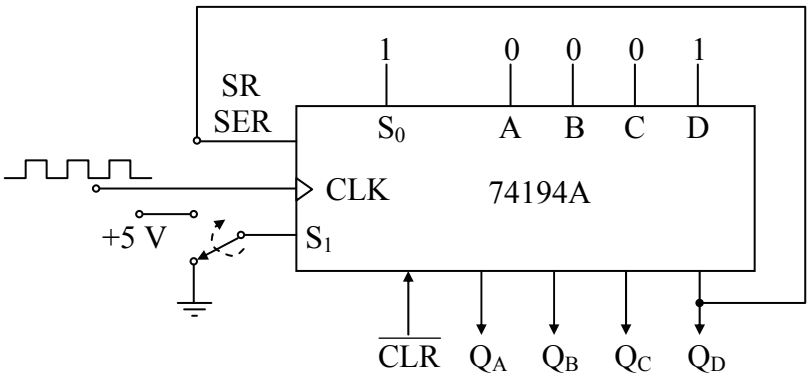



**Fig. 11.5**. Ring counter circuit

**Assignment 11.4**

**Parallel-to-serial data conversion**

1) Examine the circuit of Fig. 11.6. This circuit operates as a parallel-to-serial data converter. It first loads the data present at A through D and then shifts the data out of $Q_D$. The shifting of the loaded data is controlled by the occurrence of a START pulse. Since the START pulse occurs asynchronously to the clock pulses, the two flip-flops are used to synchronize the loading and shifting of the 74194A.

   Assume that the START pulse has been inactive and that clock pulses have been continuously applied for a long time before t0 (see the waveform in Fig. 11.6). Draw the waveforms you might expect to appear at QX, $\overline{Q}_X$, QY, and Qd in response to the START pulse shown in the figure. Use Timing Diagram 11.1.

2) Wire the circuit in Fig. 11.6. Use a normally HIGH pushbutton switch for the START pulse. Connect a square wave generator set at 100 Hz to input CLK of the 74194A. Connect toggle switches to A through D, and set these switches to 1101. Connect one vertical input of the oscilloscope to the output of the generator. Use the other vertical input to monitor first $Q_X$, $\overline{Q}_X$, $Q_Y$, and $Q_D$, in that order. Do this by pulsing the START pushbutton several times while connected to each output. You may have to slow the clock down or speed it up so that the outputs are easily observed. Verify that output waveform $Q_D$ is the serial representation for the parallel data.

   Demonstrate the circuit for your instructor.

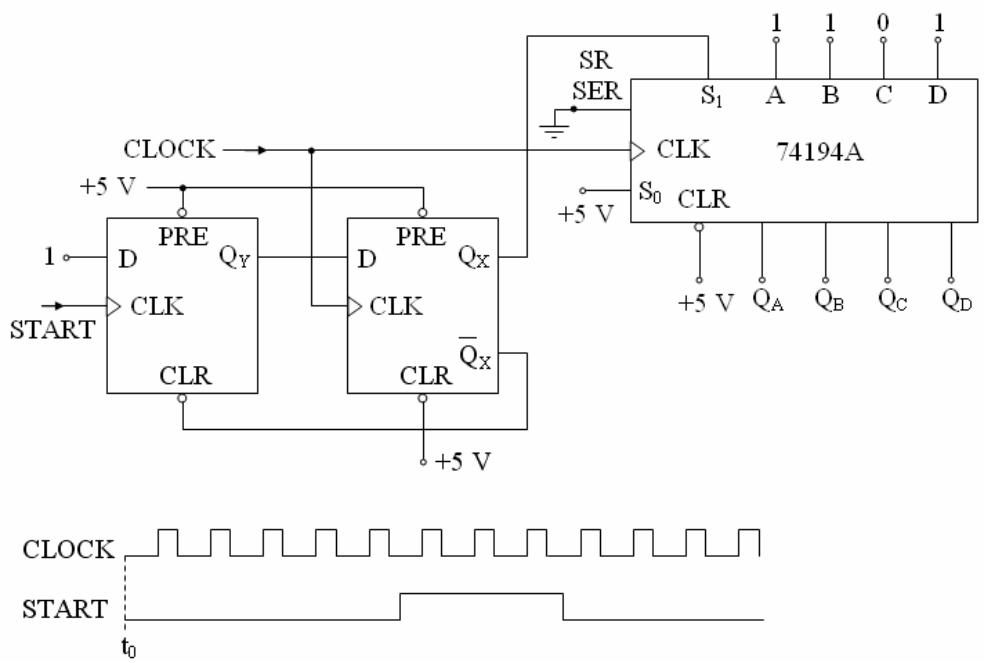



**Fig. 11.6.** Parallel-to-serial data converter


***Timing Diagram 11.1***

EXPERIMENT

# 12

# SHIFT REGISTER COUNTERS

## OBJECTIVES

1. Understand the operation and properties of ring and Johnson counters.
2. Design ring counters using flip-flops or integrated circuit registers including provisions for self-starting and self-correction.
3. Design Johnson counters using flip-flops or integrated circuit registers including provisions for self-starting and self-correction.

## EQUIPMENT REQUIRED

Digital Logic Trainer

2, 7474; 1, 7496 and 1, 74194 integrated circuits

Other integrated circuits and components, as required

## BASIC INFORMATION

Shift registers can easily be transformed into counters by feeding back some function of the stored data into the serial data input. As the register is clocked it will undergo a repetitive cycle of state transitions, i.e., it will function as a counter, depending on the type of feedback used and the initial data stored in the register.

The simplest and most widely used shift register counter is the ring counter in which the serial output is fed back to the serial input. Ring counters can be designed using flip-flops or integrated circuit registers. Fig. 12.1 shows a 3-bit ring counter designed with J-K flip-flops. Since the serial output is connected to the serial input, once the counter has been initialized to a given state, it will continue to circulate that initial data as it is clocked (if there are no glitches or noise induced state transitions – see below). Thus modulus of the counter equals the number of flip-flops. Ring counters are frequently used in applications which require sequencing a number of events. Although ting counter circuits are somewhat inefficient in the use of flip-flops, requiring one for each state, such circuits are simple, synchronous and need no additional decoding circuitry.

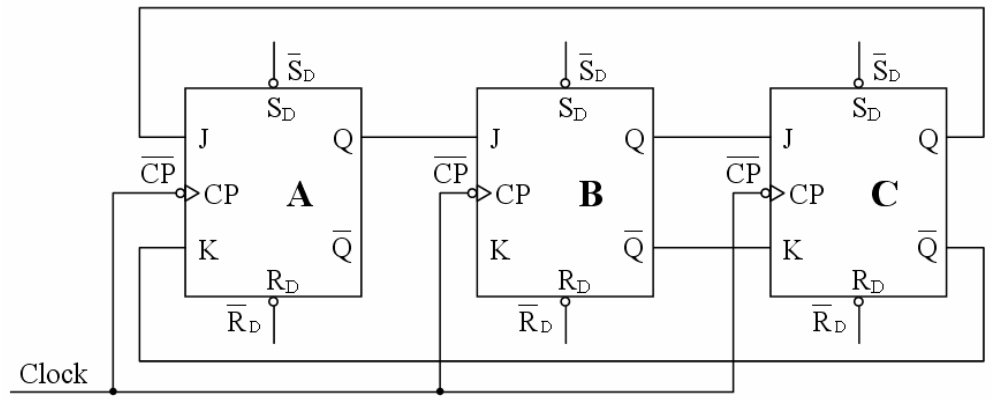



**Fig. 12.1**. Basic 3-bit ring counter circuit

There is one problem associated with ring counter operation that should not be overlooked by careful designs. For a ring counter with N flip-flops there will be N states in the normally used count sequence. However, N flip-flops have a total of $2^N$ possible states so that in an N-bit ring counter there will be $2^N$-N unused states. For the basic circuit design shown in Fig. 12.1, if the ring counter accidentally transitions into one of the unused states it will thereafter malfunction since it will no longer circulate the correct data pattern. For example, suppose the 3-bit ring counter of Fig. 12.1 were designed for active HIGH operation. The asynchronous set and reset inputs could be used to initialize the counter state to 100. Under normal operation the counter would circulate the single 1 as shown in the top portion of the state diagram. But if a glitch occurred that accidentally set another flip-flop to 1, the new data pattern would circulate as shown in the middle portion of the figure. If all flip-flops were accidentally cleared or set the counter would remain in the corresponding state indefinitely.

Fortunately, there is an easy solution to the unused state problem associated with ring counters. This is obtained by driving the serial input, not from the serial output, but rather from either the NAND or the NOR of all flip-flop outputs except for the last stage. An example of each circuit is shown in Fig. 12.2. Both circuits use a 7495 register operating in shift-right mode to form a 4-bit ring counter. First consider the circuit in Fig. 12.2 (a). Recall that the output of a NOR gate is always LOW unless all inputs are LOW. Thus the NOR gate in (a) operates as a "zero-stuffing" input to the register, continually feeding zeros into the serial data input until all flip-flops except $Q_3$ have been cleared. At this point the NOR gate output goes HIGH and a 1 will be fed into the register on the next active clock transition. Thus the circuit in Fig. 12.1 (a)

is an active HIGH ring counter with self-correcting ability. If the register accidentally transitions into an unused state it will eventually return to normal operation and will not stay locked in an unused state sequence. Note that this circuit also solves the problem of circuit initialization. When power is first applied, no matter what the initial state of the flip-flops, the register will become set to its normal state after several clock pulses. The circuit in Fig. 12.2 (b) operates in a similar fashion except the NAND gate acts as a "one-stuffer", filling the register with ones until all flip-flops but the last have been set, after which condition a zero will be fed into the serial input. Thus this circuit is an active LOW self-correcting ring counter.

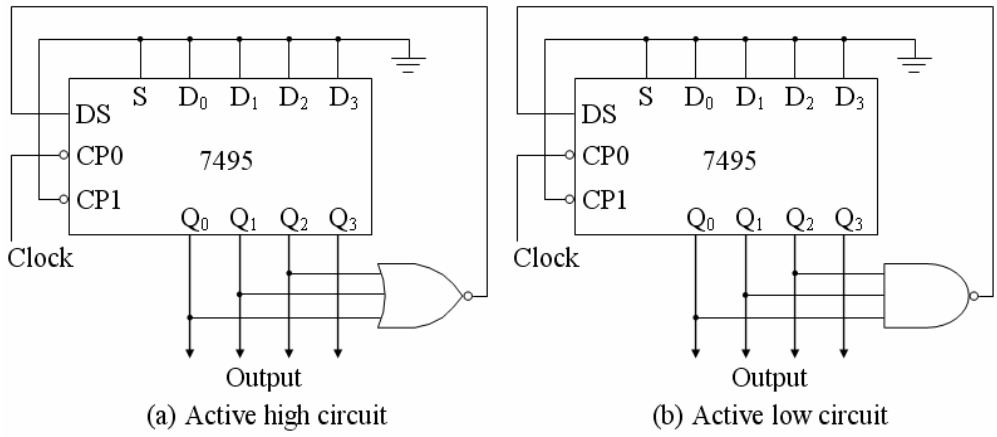



**Fig. 12.2.** Self-correcting ring counter diagrams

Another popular shift register counter is the Johnson counter. In this type of circuit the complement of the serial data output is fed back to the serial data input. A 3-bit version of this circuit using D flip-flops is shown in Fig. 12.3.

The circuit illustrates the use of a simple RC circuit to provide power-up initialization by briefly holding the direct reset lines LOW. (The values of R and C are not critical; 1 kΩ and 0.01 μF work well.) Note that there are 6 states for the 3-bit counter. In general, for an N-bit Johnson counter, there are 2N normally used states, and therefore $2^N$–2N unused states. Although there are fewer unused states than in a comparable length ring counter, they still present a problem. If the counter transitions into an unused state, it will lock into some count sequence other than that for which it was designed. This is shown in the lower half of the state diagram. The simplest way to provide self-correction for the Johnson counter is to fully decode one of the states in an unused state

sequence and use the decoded output to clear or set the register, thus returning the counter to its normal state sequence. This technique also makes the circuit self-starting.

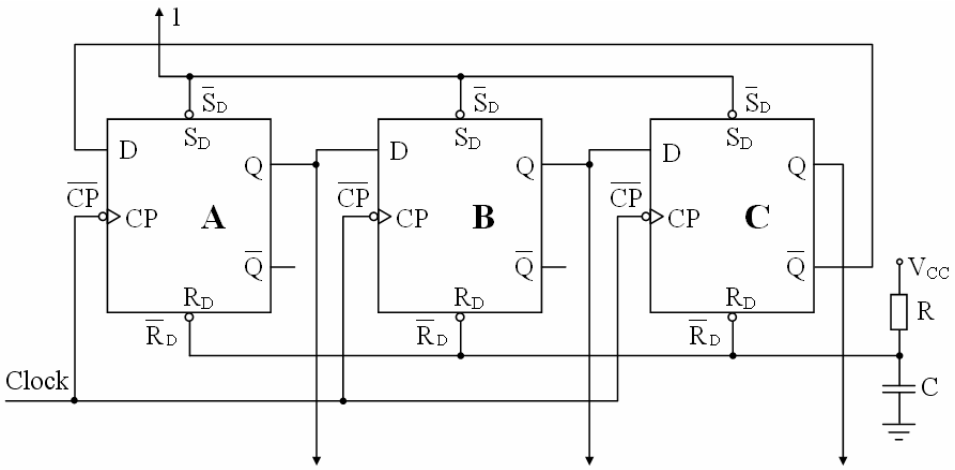



**Fig. 12.3**. 3-bit Johnson counter

One disadvantage of the Johnson counter compared to the ring counter is that the Johnson counter must include decoding circuitry if an output is required for each state. However it is always possible to decode the normally used states in a Johnson counter with 2-input AND gates. The all 1's state and the all 0's state are decoded by gating opposite ends of the flip-flop chain while the remaining states are decoded by suitably gating the adjacent 1 and 0 for each of the bit patterns.

Johnson counters represent a middle ground between ring counters and binary counters. A Johnson counter requires fewer flip-flops than a ring counter but generally more than a binary counter; it has more decoding circuitry than a ring counter but less than a binary counter. Thus, it sometimes represents a logical choice for certain applications.

## LABORATORY ASSIGNMENTS

### Assignment 12.1

1) *Ring counter*: Construct the circuit of Fig. 12.4. Connect all clear inputs to a single normally HIGH pushbutton switch. Connect the other two pushbutton switches to the DC SET and the clock input of flip-flop 3. Connect LED monitors to each flip-flop output.

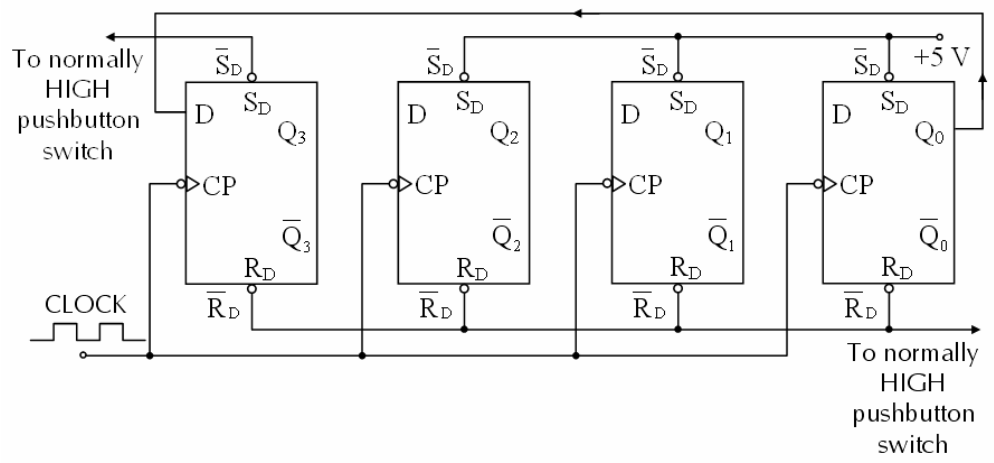



**Fig. 12.4.** Ring counter

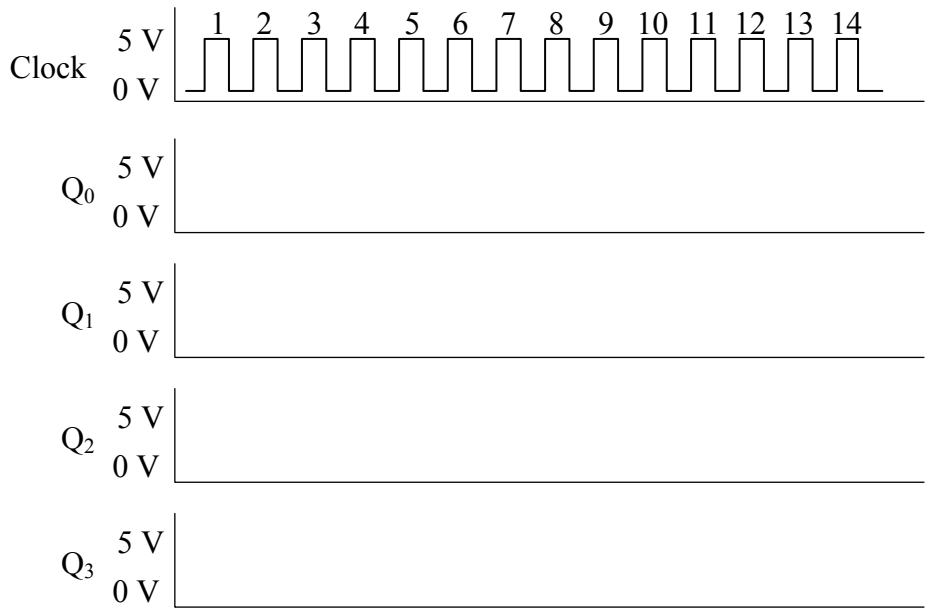
2) Clear the counter by pulsing the clear line LOW momentarily. Pulse the clock input LOW several times, and note that the counter outputs do not change. Now preset flip-flop 3 to 1 by pulsing its DC SET input HIGH momentarily. The LEDs should now indicate a count of 1000. Pulse the clock input LOW momentarily. The counter output is now ____. Pulse the clock input two more times. Observe that the 1 now occupies the rightmost position of the counter display. Now pulse the clock input once more. Where is the 1 positioned now? _____.

3) Verify that it does not matter which flip-flop is preset in order to get the counter started. Do this by reconnecting the DC SET pushbutton switch to any of the other flip-flops and repeating step 2.

4) Disconnect the clock input from the pushbutton switch, and replace it with the output of a square wave generator set at 1 kHz. Display the generator output and $Q_3$ on the oscilloscope and observe the time relationship between the two signals. Draw the waveforms on Timing Diagram 12.1. Repeat this procedure for each of the other outputs of the counter.

   The mod-number of this counter is ____. The outputs of the counter [are, are not] square waves. The frequency of each output is _____.

5) *Johnson counter:* Rewire the ring counter so that it is a Johnson counter. Disconnect the square wave generator, and reconnect the pushbutton switch to the clock input of the counter. Clear the counter. Verify the operation of the Johnson counter by pulsing the counter LOW eight times while observing the output. Record your observations in Table 12.1.




***Timing Diagram 12.1***

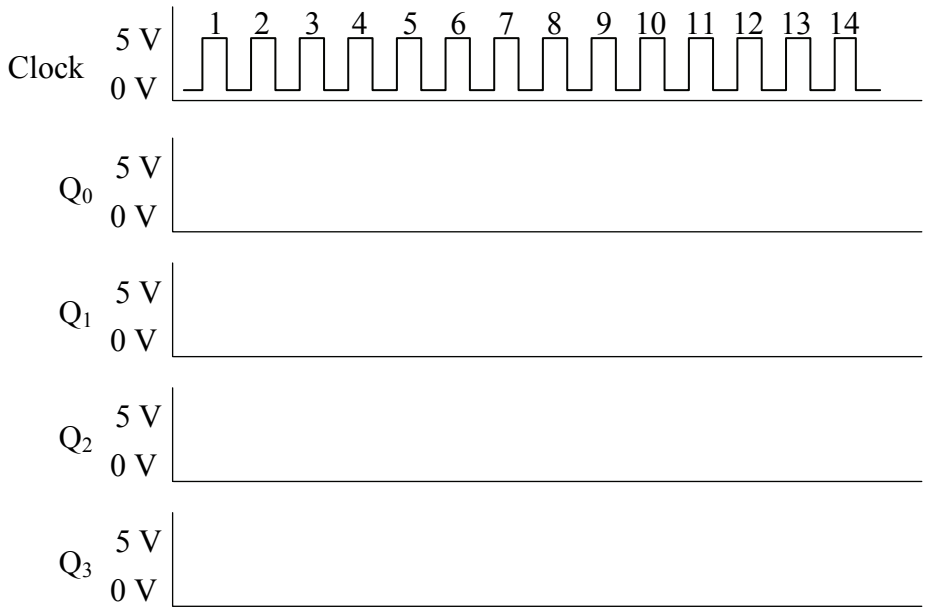
**Table 12.1**. Test results for the Johnson counter

| Shift pulse | Output state | | | |
|---|---|---|---|---|
| | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 0 | 0 | 0 | 0 | 0 |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |

6) Repeat step 4, using Timing Diagram 12.2.

The mod-number of this counter is ____. The outputs of the counter [are, are not] square waves. The frequency of each output is _____.

**Assignment 12.2**




***Timing Diagram 12.2***

Using 7474 integrated circuits, design an active HIGH self-correcting 4-bit ring counter. Build the circuit and test its operation. Use static logic level switches to control the direct set and reset inputs so that the counter can be set to any state. Use a manual pulser for the clock. Determine the circuit operation for all possible states and record the corresponding state diagram.

**Assignment 12.3**

Use a data manual to determine the properties of the 7496 5-bit shift register. Using this integrated circuit design a 5-bit active LOW self-correcting ring counter which has a control input, INIT, which when pulsed HIGH will set all flip-flops to one, so that the first clock pulse following INIT will establish the 01111 state. Build the circuit and verify the operation of the INIT control as well as the normal operating sequence of the counter.

**Assignment 12.4**

Using the 74194 integrated circuit, design a 4-bit self-correcting Johnson counter circuit which will generate the waveform shown in Fig. 12.5. Build the circuit and test its operation for the normal state sequence, confirming the

generation of the desired waveform. Using the parallel load capability of the register, force the circuit into an unused state and clock the circuit until it returns to the normal state sequence, thus confirming its self-correction ability.

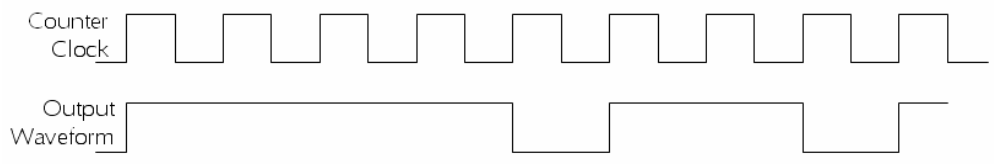



**Fig. 12.5.** Johnson counter waveforms

**REFERENCES**


1. Thomas L. Floyd, Digital Fundamentals, 9/e, Pearson education, Inc., 2006.
2. Susan A. R. Garrod, Robert J. Borns, Digital Logic: Analysis, Application and Design, Saunders College Publishing, 1991
3. M. Morris Mano, Digital Design, 2/e, Prentice-Hall, Inc., 1991
4. Ronald J. Tocci, Neal S. Widmer, Digital Systems: Principles and Applications, 7/e, Prentice-Hall, Inc., 1998
5. Jim C. DeLoach, Frank J. Ambrosio, Lab Manual (A Troubleshooting Approach) to accompany DIGITAL SYSTEMS, Principles and Applications, 6/e, Ronald J. Tocci, Prentice-Hall, Inc., 1995
6. David M. Perkins, Laboratory Manual (A Design Approach) DIGITAL ELECTRONICS, A Practical Approach, 4/e, William Kleitz, Prentice-Hall, Inc., 1996.
7. Wikipedia, The Free Encyclopedia, 7400 Series list, http://en.wikipedia.org/wiki/List_of_7400_series_integrated_circuits, 18.11.2005
8. FreeStyle Reference, Fatal Error Technical Reference, 74xxx TTL Series, http://www.fsref.com/Fatal/FE200201.SHTML, 18.11.2005.
9. J. Hewes 2006, Kelsey Park Sports College, The Electronics Club, http://www.kpsec.freeuk.com/components/74series.htm, 18.11.2005.
10. Фирма Научно-производственное предприятие ИТИС, Site Map, Cross References, Russian TTL Chips to 74-th Series, 17.01.2006 http://www.itis.spb.ru/win/tech/logic.htm
11. Semiconductor Logic Device Cross Reference, http://matthieu.benoit.free.fr/cross/competitive/logic/chipxref.htm, 21.01.06
12. Agilent Technologies, 14.2 mm (0.56 inch) Seven Segment Displays, http://www.datasheetcatalog.org/datasheet/HP/HDSP-5507.pdf, 21.01.06

## APPENDIX A

### The Breadboard

When building a "permanent circuit" the components can be "grown" together (as in an integrated circuit), soldered together (as on a printed circuit board), or held together by screws and clamps (as in house wiring). In lab, we want something that is easy to assemble and easy to change. We also want something that can be used with the same components that "real" circuits use. This is a way of making a temporary circuit, for testing purposes or to try out an idea. No soldering is required and all the components can be re-used afterwards. To build a prototype of an electronic circuit a material or device called breadboard is widely used.

The breadboard derives its name from an early form of point-to-point construction. In the early days of radio, amateurs would nail copper wire or terminal strips to a wooden board (often literally a board for cutting bread), and solder electronic components to them. Sometimes a paper schematic diagram was first glued to the board as a guide to placing terminals, components and wires.

The heart of the solderless breadboard is a small metal clip that looks like as in Fig. A.1:

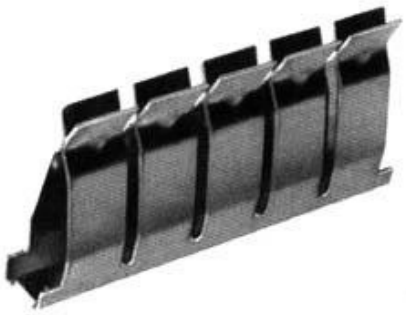


**Fig. A.1.** Socket and bus strips

The clip is made of nickel silver (which like mock turtle soup, contains no silver), a material which is reasonably conductive, reasonably springy, and reasonably corrosion resistant. Because each of the pairs of fingers is independent we can insert the end of a wire between any pair without reducing the tension in any of the other fingers. Hence each pair can hold a wire with maximum tension.

To make a breadboard, an array of these clips is embedded in a plastic block which holds them in place and insulates them from each other (Fig. A.2).

In general the breadboard consists of two terminal strips and two bus strips (often broken in the centre). Each bus strip has two rows of contacts. Each of the two rows of contacts are a node. That is, each contact along a row on a bus strip is connected together (inside the breadboard). Bus strips are used primarily for power supply connections, but are also used for any node requiring a large number of connections. Each terminal strip has 60 rows and 5 columns of contacts on each side of the centre gap. Each row of 5 contacts is a node.

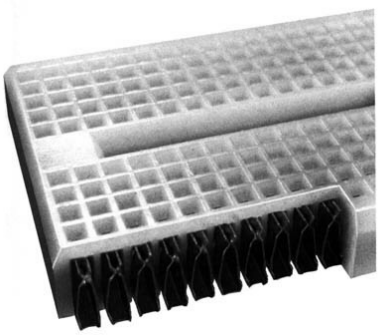


**Fig. A.2**. Embedded clips

Depending on the size and arrangement of the clips, we get either a socket strip or a bus strip. The socket strip is used for connecting components together. It has two rows of short (5 contact) clips arranged one above another (Fig. A.3).

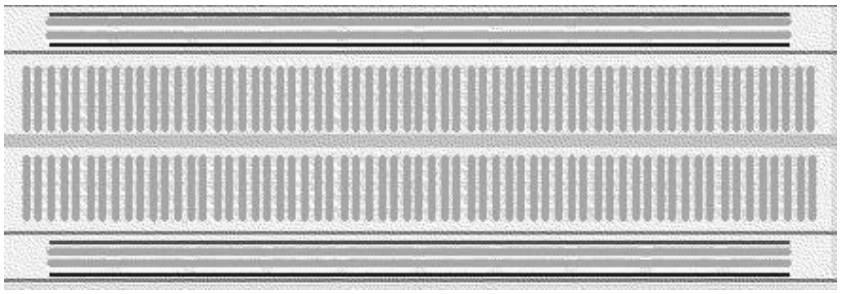


**Fig. A.3**. Shorted socket and bus strips

The bus strip is used to distribute power and ground voltages through the circuit. It has four long (25 contact) clips arranged lengthwise (Fig. A.4).

**Fig. A.4**. A bus strip

Note that in their infinite wisdom, the manufacturer elected *not* to join the adjacent 25 contact strips into a single, full-length, 50 contact strip. If this is what you want, you will have to bridge the central gap yourself.

When we combine two socket strips, three bus strips, and three binding posts on a plastic base, we get the breadboard (Fig. A.5).

These strips connect the holes on the top of the board. This makes it easy to connect components together to build circuits. To use the breadboard, the legs of components are placed in the holes (the sockets). The holes are made so that they will hold the component in place. Each hole is connected to one of the metal strips running underneath the board.

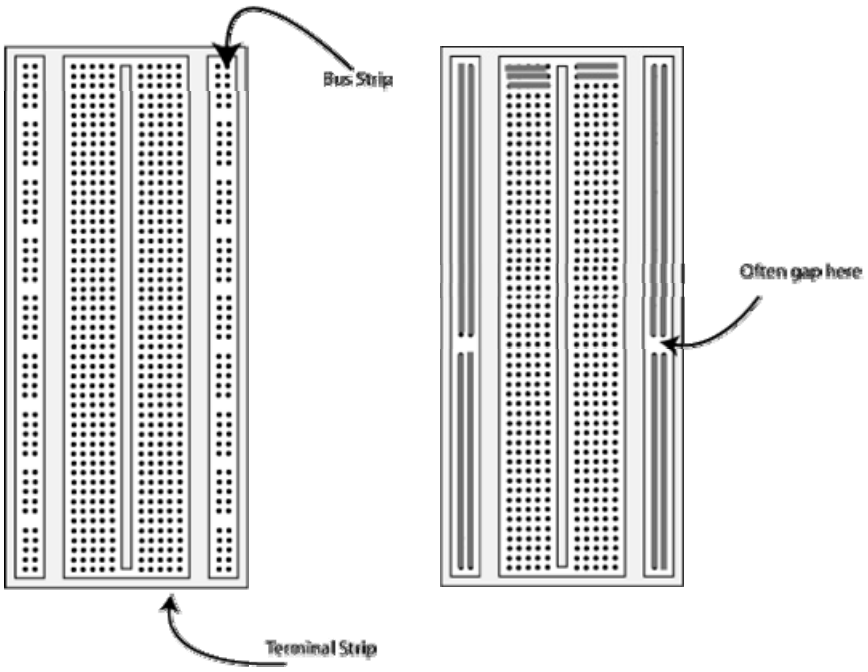



**Fig. A.5.** The breadboard. The orange lines indicate connected holes

Each wire forms a node. A node is a point in a circuit where two components are connected. Connections between different components are formed by putting their legs in a common node. On the bread board, a node is the row of holes that are connected by the strip of metal underneath.

The long top and bottom row of holes are usually used for power supply connections.

The rest of the circuit is built by placing components and connecting them together with jumper wires. Then when a path is formed by wires and components from the positive supply node to the negative supply node, we can turn on the power and current flows through the path and the circuit comes alive.

You will build your circuits on the terminal strips by inserting the leads of circuit components into the contact receptacles and making connections with 22-26 gauge wire. There are wire cutter/strippers and a spool of wire in the lab. It is a good practice to wire +5 V and 0 V power supply connections to separate bus strips.

The 5V supply ***MUST NOT BE EXCEEDED*** since this will damage the integrated circuits (ICs) used during the experiments. Incorrect connection of power to the ICs could result in them exploding or becoming very hot - with the ***possible serious injury occurring to the people working on the experiment!***

**Building the Circuit**

Throughout these experiments we will use TTL chips to build circuits. The steps for wiring a circuit should be completed in the order described below:

- Turn the power off before you build anything!
- Make sure the power is off before you build anything!
- Connect the +5 V and ground (GND) leads of the power supply to the power and ground bus strips on your breadboard. Before connecting up, use a voltmeter to check that the voltage does not exceed 5 V.
- Plug the chips you will be using into the breadboard. Point all the chips in the same direction with pin 1 at the upper-left corner. (Pin 1 is often identified by a dot or a notch next to it on the chip package).
- Connect +5 V and GND pins of each chip to the power and ground bus strips on the breadboard.
- Select a connection on your schematic and place a piece of hook-up wire between corresponding pins of the chips on your breadboard. It is better to make the short connections before the longer ones. Mark each connection on your schematic as you go, so as not to try to make the same connection again at a later stage.
- Get one of your group members to check the connections, **before you turn the power on**.
- If an error is made and is not spotted before you turn the power on. Turn the power off immediately before you begin to rewire the circuit.

- At the end of the laboratory session, collect you hook-up wires, chips and all equipment and return them to the demonstrator.
- Tidy the area that you were working in and leave it in the same condition as it was before you started.

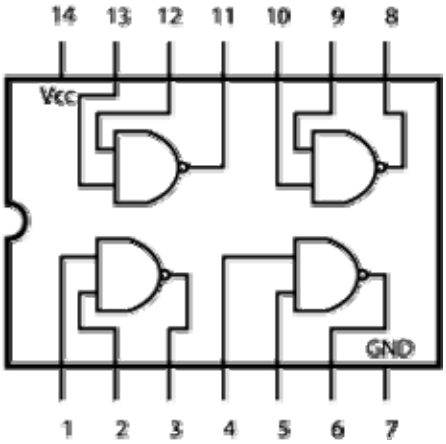
**Common Causes of Problems**

- Not connecting the ground and/or power pins for all chips.
- Not turning on the power supply before checking the operation of the circuit.
- Leaving out wires.
- Plugging wires into the wrong holes.
- Driving a single gate input with the outputs of two or more gates.
- Modifying the circuit with the power on.

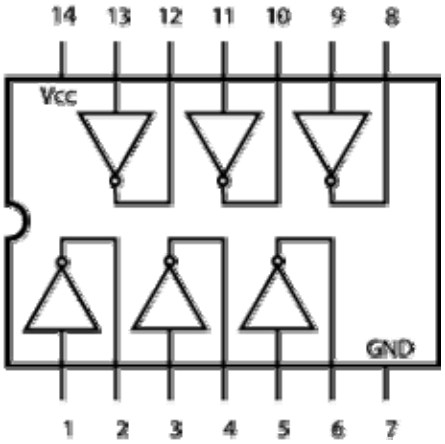
In all experiments, you will be expected to obtain all instruments, leads, components at the start of the experiment and return them to their proper place after you have finished the experiment. Please inform the demonstrator or technician if you locate faulty equipment. If you damage a chip, inform a demonstrator, don't put it back in the box of chips for somebody else to use.

**Example Implementation of a Logic Circuit**

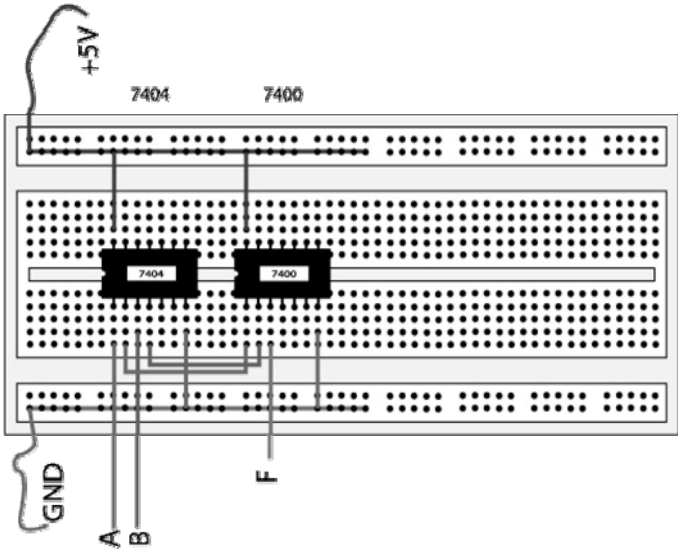
Build a circuit to implement the Boolean function $F = \overline{\overline{A} \cdot \overline{B}}$.




*Quad 2 Input 7400*



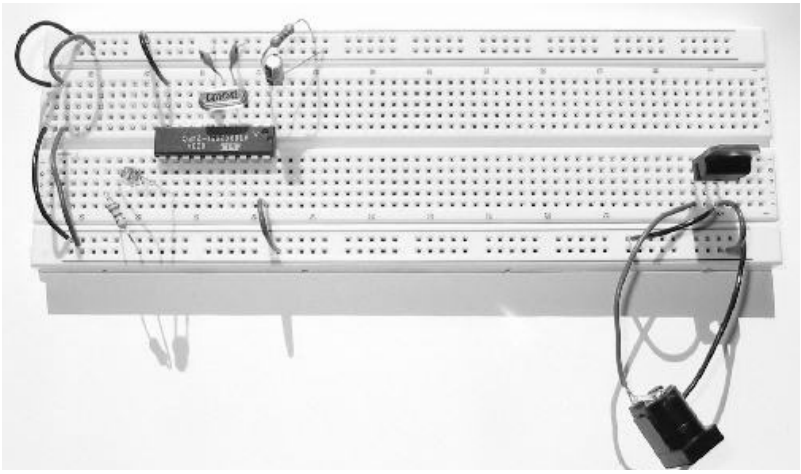

*Hex 7404 Inverter*




***The complete designed and connected circuit***

Sometimes the chip manufacturer may denote the first pin by a small indented circle above the first pin of the chip. Place your chips in the same direction, to save confusion at a later stage. Remember that you must connect power to the chips to get them to work.

For chips with many legs (ICs), place them in the middle of the board so that half of the legs are on one side of the middle line and half are on the other side. A completed circuit might look like the following:



***A completed circuit assembled on the breadboard***

# APPENDIX B

## WIRING AND TROUBLESHOOTING DIGITAL CIRCUITS

### Objectives

1. To discuss general wiring procedures for digital circuits.
2. To introduce the student to formalized troubleshooting procedures.
3. To list some of the common faults found in digital systems.

### Discussion

The experiments in this manual are designed to give you hands-on experience with digital circuits. More than that, they provide you with an opportunity to develop sound breadboarding and troubleshooting skills that will be invaluable to you whether you eventually become an engineer or a technician. This appendix will present some very basic information and suggestions concerning each area. It is not meant to replace any laboratory standards. However, much of the information given here can be used as reference material that can be, and should be, reviewed from time to time.

Most of the experiments contain operational testing of various ICs. At times, this may appear to be a tedious undertaking on your part. Don't fall into the trap of treating this sort of experimentation mechanically, taking for granted that an IC will operate just as it did in the classroom lecture. In the classroom, you are working with the ideal. In the lab, you will occasionally work with ICs that are less than ideal. In fact, they may not work at all, or at least not in the manner they were designed to work. If you keep in mind that lab experimentation is not only to verify principles but also to learn to recognize common problems associated with the circuits, you will get more out of the experiments. As you will learn, verification of a circuit's operation is one of the first steps taken in troubleshooting.

### B.1. Prototype Circuit Wiring

It is assumed that you will be wiring circuits using a prototype circuit board. Such boards come in different sizes, but most have the following features:

a) Two horizontal rows of holes, one at the top and one at the bottom. The contacts underneath the holes on each of these rows are connected together to form a bus. They are not directly connected to the other holes on the board.

b) At least two sections of holes, with each section arranged so that the holes are in vertical groups called circuit blocks. Each circuit block is isolated from all others. This permits several wires to be joined at common junctions. The two sections are separated by a horizontal gap. This gap separates the sections electrically as well as physically. Thus, a vertical circuit block in the top section of the board is not connected to the block directly below it in the bottom section. ICs will straddle this gap so that each IC pin will be inserted into its own block. Connections to each pin will be brought to its block.

*Installing ICs:* ICs should be installed or mounted on the board to permit wires going from the top section of the board to the bottom to go between the ICs. It is not advisable to pass wires over ICs, although sometimes it is hard to avoid. Strapping ICs to the board in this manner will present problems if the IC has to be removed. The consequences of this are obvious.

As you mount an IC, check to make sure that none of its pins are being tucked beneath it. If it is necessary to remove an IC, always use an IC puller. Never remove an IC with your fingers or with a pair of pliers. The first causes a definite safety hazard, while the second will often result in eventual damage to the IC.

*Wiring the circuit:* Wires should be dressed so that 3/8" insulation is stripped from each end and the length of wire is no more than needed to make a neat connection between circuit blocks. If the wires are too long, some circuits will malfunction, especially flip-flops and flip-flop devices such as counters. You may have to rearrange the ICs on the board to solve this problem, if it occurs. Another way to solve the problem is by inserting a 2 kΩ resistor in series with the wire at the input end of the wire.

Have a lab partner call out each connection to be made. Route the wires along the circuit board neatly, bending them smoothly wherever necessary. Avoid bending the wire sharply, since this will increase the likelihood of fracture beneath the insulation, resulting in an open circuit or an intermittent open. Minimize the number of crossovers, that is, wires routed over other wires.

The overall appearance should be neat, not like a bowl of spaghetti. If you have made all of your connections as outlined above, it may not be picture perfect, but the neatness will pay off in reduced troubleshooting time and easier IC replacement.

### B.2. Testing the Circuit

Circuit testing is also known as troubleshooting. You are probably accustomed to discrete circuit (e.g., a transistor amplifier) troubleshooting methods. Since

each circuit element of a discrete circuit is accessible to the troubleshooter, faulty circuit elements can be isolated by making basic measurements such as voltage, resistance, capacitance, and inductance, using conventional test equipment. Modern digital circuits and systems, on the other hand, consist mainly of digital ICs. The IC's components are not accessible to the troubleshooter, so the troubleshooter must rely on knowledge of the IC's operation(s) in order to isolate the IC as being faulty. The experiments in this manual are designed to give you the necessary experience to test for and recognize proper operation of ICs.

A digital IC is considered defective or faulty if its outputs do not respond correctly, according to its truth table, for each set of input conditions and for each of its various operating modes. A similar statement can be made for digital circuits and systems. Once it has been verified that a circuit or system is not responding correctly, a *fault is said to exist*, and further troubleshooting is indicated. The next troubleshooting step is to *isolate the cause of the fault*, which may be in one or more smaller circuits or subsystems. By progressively isolating smaller circuits, and perhaps smaller subsystems, the troubleshooter will eventually isolate the defective components, which may be one or more ICs and/or discrete components. After *replacing the defective components*, the circuit or system is tested for proper operation once more. Once proper operation is established for all operating modes, the troubleshooter's task is completed.

Now that the student is acquainted with the nature of digital troubleshooting, a procedure for fault isolation is presented. The student should, when applying the procedure in the lab, perform each step in the order given. After sufficient experience with digital circuits is attained, common sense and intuition may lead the student directly to the faulty device and thereby reduce the amount of troubleshooting time.

*Step 1:* Perform a visual inspection of the system or circuit. Look for loose or damaged connecting wires, cables, and printed circuit (PC) boards, evidence of burning or extreme overheating, missing components, and blown fuses. If the circuit or system is mounted on prototype boards, look for wiring errors, damaged boards, and digital ICs improperly inserted. Also check for incorrect circuit design.

*Step 2:* Check all power source levels, and confirm that power is actually being applied to the circuit or system.

*Step 3:* Study all relevant documentation on the circuit or system, such as block diagrams, schematics, and operating instructions. Learn how the circuit or system operates normally.

*Step 4:* Verify all operating modes of the circuit by running tests.

*Step 5:* Record results of the tests run in step 4. Test results often show patterns that may lead to the faulty device. Repeat steps 4 and 5 at least once before proceeding to step 6.

*Step 6:* If the circuit passes all tests, end the procedure. If the circuit fails at least one test, continue to the next step.

*Step 7:* Analyze the test results recorded above and select a possible location for the fault.

*Step 8:* Check all signals and static logic levels at this location, and record them. If nothing appears abnormal, return to step 7.

*Step 9:* Analyze the test results recorded above, and select a possible faulty device.

*Step 10:* Check the device for proper functioning. If it is a discrete component, take basic Ohm's Law measurements and/or use a device tester to determine if the device is faulty. If the device is an IC, check the IC for proper functioning. This includes checking inputs and outputs for stuck-HIGH and stuck-LOW conditions and other types of digital IC faults (see below, *Common Digital IC Faults*). If the device passes all tests, then return to step 9.

*Step 11:* Repair or replace the faulty device and return to step 4.

### B.3 Common Causes of Faults in Digital Systems

In this section, several common causes of faults in digital systems are listed along with symptoms given for each cause and steps that may be taken to correct or minimize its effects on the system.

*Defective components:* Components normally fail because of age, because the maximum voltage or current rating of the device was exceeded due to improper design or because of the breakdown of another component, improper connections, or excessive ambient temperature. In the case of digital ICs, overheating caused by improper connections (especially prototype circuits), overvoltage, or ambient temperature may result in the IC operating only sporadically. After cooling down, the IC will usually operate normally.

*IC loading problems:* Exceeding the fan-out of a TTL logic output may result in the output voltage dropping below $V_{OH(min)}$ or rising above $V_{OL(max)}$. To verify this condition, the output voltage should be checked for a level of 0 V – 0.8 V for a LOW and 2 V – 5 V for a HIGH. If not, the excessive fan-out is causing a problem.

CMOS and MOS logic outputs will not be affected significantly by exceeding the fan-out limits. However, any transitions at the output will show an increase in rise time and fall time. This is because each CMOS or MOS input loads the output capacitively (5 pF each input).

Some common symptoms to look for are flip-flops, counters, and flip-flop registers that do not respond to the signal at the clock input. Measure $t_R$ and $t_F$ of the clock signal, and compare the measurements to the minimum required by the flip-flop, counter, or register for proper triggering.

To correct the problem caused by excessive fan-out, a buffer should be used or the fan-out reduced by load splitting. Another solution is to insert a pulse-shaping circuit such as a Schmitt trigger between the overloaded output and the clock input.

*Improper signal characteristics:* A digital IC may function improperly if logic signals not meeting its requirements are applied to its inputs. Minimum requirements are given for amplitude, pulse duration, and transition times. A signal that fails to meet anyone of these requirements can cause the IC to function incorrectly.

Common symptoms brought on by improper signal characteristics include flip-flops, counters, and flip-flop registers that respond incorrectly to signals at clock, clear, and preset inputs.

The characteristic(s) causing the problem must be determined and brought back into specification.

*Power supply-Improper levels:* Since all IC logic devices use voltage to represent logic levels, trouble with the output level of the supply can cause ICs to function improperly. A common cause of improper power supply levels is overload. This can be particularly true in prototype systems and circuits. ..

Symptoms of this type of trouble include the condition where logic HIGH at circuit outputs is less than $V_{OH}$. Disconnecting a few ICs from the power supply will usually cause the level of $V_{CC}$ to rise if this is the case.

Using a larger power supply or redesigning the existing one for higher current output will solve the problem.

*Power supply-Poor regulation*. Poor regulation in a power supply will cause $V_{CC}$ to fluctuate when large numbers of logic circuits are switching states. These fluctuations act like noise pulses and can cause false triggering of logic devices. This problem is especially significant in TTL circuits.

A symptom caused by this problem is flip-flops, counters, and registers triggering when they are not supposed to, and triggering instead at the time other devices in the system are changing states. To verify that poor regulation is the problem, $V_{CC}$ should be examined with an oscilloscope. If spikes or pulses are riding on the $V_{CC}$ level causing $V_{CC}$ to drop by more than 0.2 V, then the power supply has poor regulation.

There are two ways to correct this problem: (1) improve the power supply regulation by either replacing or redesigning the current one, or (2) use RF decoupling capacitors.

*Grounding problems:* Poorly designed ground return circuits can cause the voltage at IC ground pins to be nonzero. This is because currents flowing through the ground system can cause resistive and inductive voltage drops (see Fig. B.1). To avoid this problem, all ground wires should have low resistance and inductance, and each IC ground pin should be connected to the power supply separately. PC board ground returns should be large conductive traces.

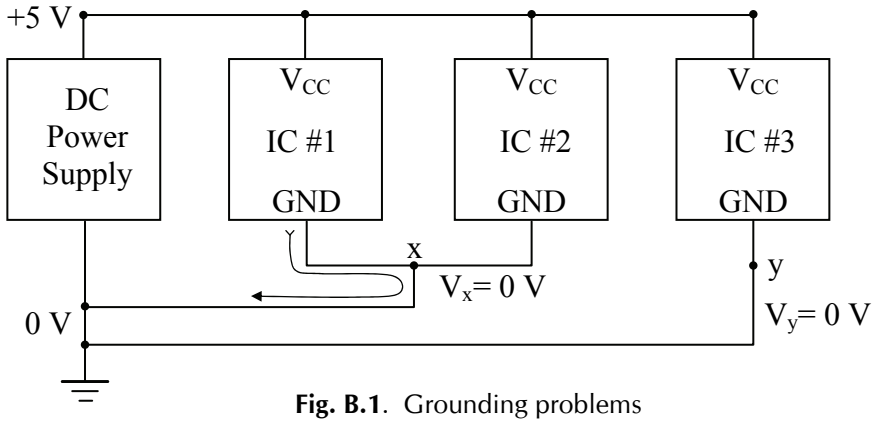



**Fig. B.1**. Grounding problems

*Noise problems:* Circuit noise can be externally or internally generated. Internally generated noise was discussed earlier. Externally generated noise can cause sporadic triggering of logic circuits. Common sources are electro-mechanical devices (e.g., motors and relays that produce electromagnetic radiation) and electronic power control circuitry using SCRs and TRIACs. This type of problem can be minimized by using special AC power line filtering devices to prevent noise from entering through the AC lines and grounded shields or conducting planes to short radiated noise signals to ground.

### B.4 Common Digital IC Faults

Digital IC faults are classified as either internal or external faults. We begin our discussion with internal faults.

*Internal digital IC faults:* There are four types of internal failures:

1) inputs or outputs shorted to ground or $V_{CC}$
2) inputs or outputs open
3) shorts between pins (not to ground or $V_{CC}$)
4) internal circuitry failure

These failures are corrected by replacing the faulty IC. A discussion on each type of failure follows.

*Short to ground or $V_{CC}$:* This failure causes the inputs or outputs to be either permanently HIGH or permanently LOW (referred to as stuck-HIGH or stuck-LOW). Fig. B.2(a) shows a NAND gate with a stuck-LOW input and a stuck-HIGH output. The stuck-HIGH condition may be the result of an internal short in input A, an internal short at output X, or both.

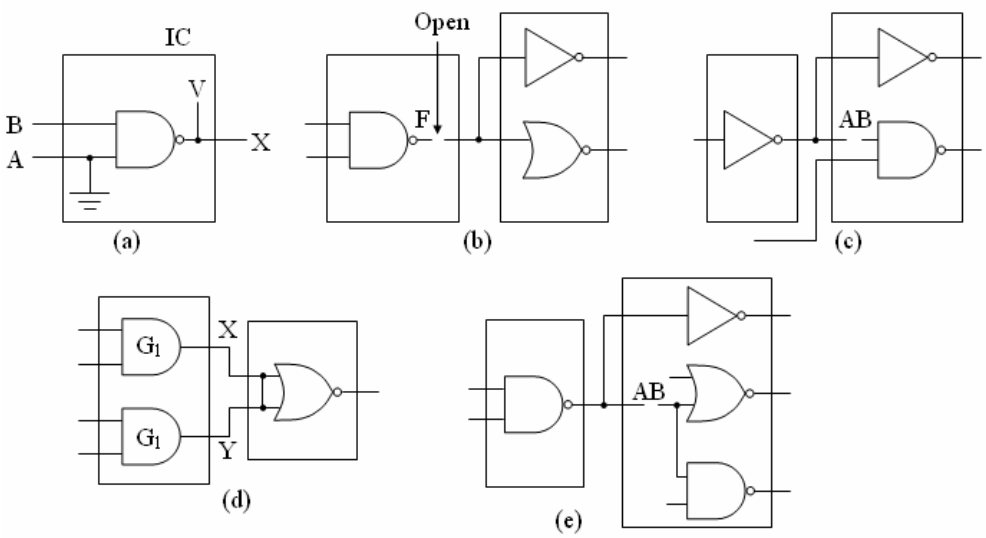



**Fig. B.2**. Types of failure

Connections to output X are also forced HIGH; connections to input A are forced LOW. Shorts of this type in emitter-coupled logic (ECL) devices result in neither a HIGH nor a LOW.

In troubleshooting this type of failure, the student should be aware that signals may not change beyond the point where the short is located.

*Open inputs or outputs:* An open output will result in an open input for all inputs driven by the output. Open inputs in TTL logic devices generally act as HIGHs, causing inputs that are tied to open outputs to resemble a stuck-HIGH input, though not always. Open CMOS inputs do not generally act as a HIGH or a LOW. This being the case, inputs tied to an open TTL or CMOS output will resemble a stuck-LOW or stuck-HIGH input or may even oscillate between HIGH and LOW. Open inputs for ECL devices, with inputs pulled down by a resistor, are LOW.

Fig. B.2(b) illustrates an open output in a NAND gate. Fig. B.2(c) shows an open input. In the latter diagram, the student should note that all signals before point A are unaffected.

*Short between two pins:* Fig. B.2(d) shows two input pins shorted together. This means that the outputs of the two driver gates are also shorted. This condition will cause a fault in TTL and CMOS devices only if the two driver outputs try to go to opposite levels, say, X to HIGH and Y to LOW. In this case, if the device is TTL, X will be stuck-LOW. In other words, if one output is LOW, both will be LOW. However, if the device is CMOS, this condition typically produces an intermediate level (see Fig. B.3). There is obviously no fault which occurs when both outputs are supposed to be at the same level. ECL device outputs can normally be connected together, so no logic faults will occur unless the driver gates are damaged by excessive currents.

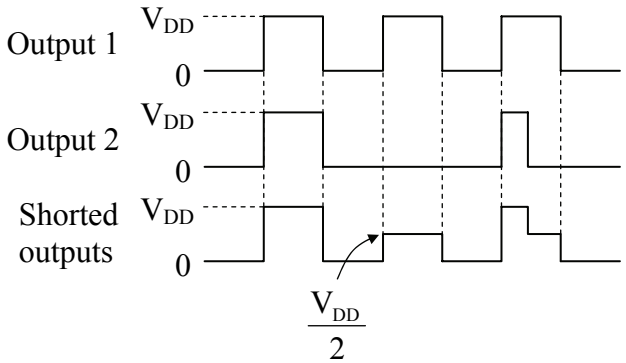



**Fig. B.3**. CMOS: short between pins and intermediate levels

*Internal circuitry failure:* Failure in the circuits within a digital IC can cause its inputs and outputs to be stuck-HIGH or stuck-LOW.

*External digital IC faults:* In addition to the four types of internal failure, there are four types of external failures that can occur:

1) line shorted to ground or power supply
2) open signal line
3) short between signal lines
4) failure of a discrete component

To discover these faults, look for poor soldering joints, solder bridges, open wires or traces, or test components such as capacitors and resistors for opens, shorts, and/ or values that are out of tolerance.

*Line shorted to ground or power supply:* This type of failure will appear like an internal short and can't be distinguished from it. Perform a careful visual inspection to isolate this fault.

*Open signal line:* Fig. B.2(e) shows an open signal line that results in an open input only for points beyond point B. All inputs before A are unaffected by the open line. Signal tracing and/ or continuity checks are useful techniques for discovering this type of fault.

*Short between two signal lines:* This type of fault cannot be distinguished from an internal short. Often, poor soldering on PC boards results in solder bridges across the signal lines. On prototype boards, look for bare connecting wires (poorly dressed) too close together. In either case, a visual inspection is necessary to locate this fault.

Shorted signal lines in TTL will appear different from shorts in CMOS circuits. In TTL, if one signal is trying to go HIGH and the other is going LOW, the level at the short will be about a V. This is because resistance at TTL outputs is lower in the LOW state than in the HIGH state. For CMOS and MOS devices, the level at the short will be about midway between a v and 5 V for this same situation, because their output resistance is about the same in both states. See Fig. B.3 for an example of how waveforms would look for shorted signal lines in CMOS and MOS circuits. Note the 2.5 V levels. These levels would not normally appear on the waveforms.

*Failure of discrete components:* While most digital components are ICs, there is still circuitry that requires discrete components such as resistors, capacitors, transistors, and diodes. These components can be tested either completely out of the circuit or by unsoldering one or more of their leads and checking them with an appropriate test instrument such as an ohmmeter, capacitance checker, or transistor checker. Faulty discrete components could mean another circuit caused the failure. Be sure and check around for other faults because, in the long run, this avoids repeated failures in the device replaced.

*Common test equipment used in digital troubleshooting:* Besides the usual analog test equipment, such as VOMs, oscilloscopes, and the like, digital troubleshooting requires some specialized equipment. A list of these specialized instruments would include the following:

1) logic probe
2) logic pulser
3) current tracer
4) logic analyzer

Of the four, the logic probe is the most useful in general troubleshooting. The pulser is useful when it is necessary to trigger gates, flip-flops, counters, or other types of circuits to check for proper operation.

The current tracer is a more specialized test probe used in locating shorts in digital circuits. Whenever a short circuit is suspected, the current tracer can assist the troubleshooter in pinpointing the exact location of the short.

The logic analyzer is a complex instrument used to compare many different logic signals at one time. However, it is expensive and is used mostly in complex systems to solve the more difficult problems that occur in digital systems.

The experiments in this manual provide opportunities to gain experience with the logic probe, logic pulser, and the logic analyzer. It is recommended that you become acquainted with the logic probe you will be using by reading the user manual that should accompany the probe. If your laboratory has a logic analyzer, it would also be to your advantage to learn as much as possible about the instrument before you attempt to use it. If there is an operator's manual for the analyzer, get it and read it.

**Concluding Remarks**

The material in this appendix will be of more use to you if you review it from time to time. There is too much information relating to troubleshooting to include all of it in a short appendix. Your learning resource center or library may have some video tapes, journals, or books on the subject.

# APPENDIX C

## 74 SERIES ICs and CROSS-REFERENCE of 155 SERIES RUSSIAN ICs

### General characteristics

There are several families of logic chips numbered from 74xx00 onwards with letters (xx) in the middle of the number to indicate the type of circuitry, e.g. 74LS00 and 74HC00. The original family (now obsolete) had no letters, e.g. 7400.

The **74LS** (Low-power Schottky) family (like the original) uses TTL (Transistor-Transistor Logic) circuitry which is fast but requires more power than later families. The 74 series is often still called the "TTL series" even though the latest chips do not use TTL!

The **74HC** family has high-speed CMOS circuitry, combining the speed of TTL with the very low power consumption of the 4000 series. They are CMOS chips with the same pin arrangements as the older 74LS family. Note that 74HC inputs cannot be reliably driven by 74LS outputs because the voltage ranges used for logic 0 are not quite compatible, use 74HCT instead.

The **74HCT** family is a special version of 74HC with 74LS TTL-compatible inputs so 74HCT can be safely mixed with 74LS in the same system. In fact 74HCT can be used as low-power direct replacements for the older 74LS ICs in most circuits. The minor disadvantage of 74HCT is a lower immunity to noise, but this is unlikely to be a problem in most situations.

For most new projects the 74HC family is the best choice.

The 74LS and 74HCT families require a 5V supply so they are not convenient for battery operation.

### 74HC and 74HCT family characteristics:

- **74HC Supply**: 2 to 6 V, small fluctuations are tolerated.
- **74HCT Supply**: 5 V $\pm$0.5 V, a regulated supply is best.
- **Inputs** have very high impedance (resistance), this is good because it means they will not affect the part of the circuit where they are connected. However, it also means that unconnected inputs can easily pick up electrical noise and rapidly change between high and low states in an unpredictable way. This is likely to make the chip behave erratically and it

will significantly increase the supply current. To prevent problems all unused inputs MUST be connected to the supply (either +Vs or 0V), this applies even if that part of the chip is not being used in the circuit!

**Note that 74HC inputs cannot be reliably driven by 74LS outputs** because the voltage ranges used for logic 0 are not quite compatible. For reliability use **74HCT** if the system includes some 74LS chips.

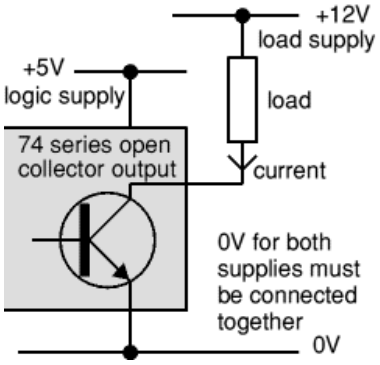
- **Outputs** can sink and source about 4 mA if you wish to maintain the correct output voltage to drive logic inputs, but if there is no need to drive any inputs the maximum current is about 20 mA. To switch larger currents you can connect a transistor.
- **Fan-out**: one output can drive many inputs (50+), except 74LS inputs because these require a higher current and only 10 can be driven.
- **Gate propagation time**: about 10 ns for a signal to travel through a gate.
- **Frequency**: up to 25 MHz.
- **Power consumption** (of the chip itself) is very low, a few $\mu$W. It is much greater at high frequencies, a few mW at 1 MHz for example.

**74LS family TTL characteristics:**

- **Supply**: 5 V $\pm$0.25 V, it must be very smooth, a regulated supply is best. In addition to the normal supply smoothing, a 0.1 $\mu$F capacitor should be connected across the supply near the chip to remove the "spikes" generated as it switches state, one capacitor is needed for every 4 chips.
- **Inputs** "float" high to logic 1 if unconnected, but do not rely on this in a permanent (soldered) circuit because the inputs may pick up electrical noise. 1mA must be drawn out to hold inputs at logic 0. In a permanent circuit it is wise to connect any unused inputs to +Vs to ensure good immunity to noise.
- **Outputs** can sink up to 16mA (enough to light an LED), but they can source only about 2 mA. To switch larger currents you can connect a transistor.
- **Fan-out**: one output can drive up to 10 74LS inputs, but many more 74HCT inputs.
- **Gate propagation time**: about 10 ns for a signal to travel through a gate.
- **Frequency**: up to about 35 MHz (under the right conditions).
- **Power consumption** (of the chip itself) is a few mW.

### Open Collector Outputs

Some 74 series ICs have open collector outputs, this means they can sink current but they cannot source current. They behave like an NPN transistor switch.




The diagram shows how an open collector output can be connected to sink current from a supply which has a higher voltage than the logic IC supply. The maximum load supply is 15 V for most open collector ICs.

Open collector outputs can be safely connected together to switch on a load when any one of them is low; unlike normal outputs which must be combined using diodes.

There are many ICs in the 74 series and this page only covers a selection, concentrating on the most useful gates, counter, decoders and display drivers. For each IC there is a diagram showing the pin arrangement and brief notes explain the function of the pins where necessary. For simplicity the family letters after the 74 are omitted in the diagrams below because the pin connections apply to all 74 series gates with the same number. For example 7400 NAND gates are available as 74HC00, 74HCT00 and 74LS00.

If you are using another reference please be aware that there is some variation in the terms used to describe pin functions, for example reset is also called clear. Some inputs are "active low" which means they perform their function when low. If you see a line drawn above a label it means it is active low, for example: $\overline{\text{reset}}$ (say "reset-bar").

Here is a comprehensive cross-reference of TTL and CMOS chips that are readily available.

Tables of both TTL and CMOS devices are provided along with tables grouping chips with the same functionality together.

### Logic ICs (chips)

Logic ICs process digital signals and there are many devices, including logic gates, flip-flops, shift registers, counters and display drivers. They can be split into two groups according to their pin arrangements: the 4000 series and the 74 series which consists of various families such as the 74HC, 74HCT and 74LS.

**For most new projects the 74HC family is the best choice**. The older 4000 series is the only family which works with a supply voltage of more than 6V. The 74LS and 74HCT families require a 5V supply so they are not convenient for battery operation.

The table below summarises the important properties of the most popular logic families:

| **Property** | **4000 Series** | **74 Series 74HC** | **74 Series 74HCT** | **74 Series 74LS** |
|---|---|---|---|---|
| **Technology** | CMOS | High-speed CMOS | High-speed CMOS TTL compatible | TTL Low-power Schottky |
| **Power Supply** | 3 to 15 V | 2 to 6 V | 5 V ± 0.5 V | 5 V ± 0.25 V |
| **Inputs** | Very high impedance. Unused inputs must be connected to +Vs or 0 V. Inputs cannot be reliably driven by 74LS outputs unless a 'pull-up' resistor is used. | | Very high impedance. Unused inputs must be connected to +Vs or 0 V. Compatible with 74LS (TTL) outputs. | 'Float' high to logic 1 if unconnected. 1mA must be drawn out to hold them at logic 0. |
| **Outputs** | Can sink and source about 5 mA (10 mA with 9 V supply), enough to light an LED. To switch larger currents use a transistor. | Can sink and source about 20 mA, enough to light an LED. To switch larger currents use a transistor. | Can sink and source about 20 mA, enough to light an LED. To switch larger currents use a transistor. | Can sink up to 16 mA (enough to light an LED), but source only about 2 mA. To switch larger currents use a transistor. |
| **Fan-out** | One output can drive up to 50 CMOS, 74HC or 74HCT inputs, but only one 74LS input | One output can drive up to 50 CMOS, 74HC or 74HCT inputs, but only 10 74LS inputs. | | One output can drive up to 10 74LS inputs or 50 74HCT inputs. |

| **Maximum Frequency** | about 1 MHz | about 25 MHz | about 25 MHz | about 35 MHz |
|---|---|---|---|---|
| **Power consumption** of the IC itself | A few $\mu$W | A few $\mu$W | A few $\mu$W | A few mW |

### TTL Device Summary

| Device | Function | Similar Devices |
|---|---|---|
| 7400 | Quad 2-Input NAND Gate | 7403/74132/4011/4093/40107 |
| 7402 | Quad 2-Input NOR Gate | 4001 |
| 7403 | Quad 2-Input NAND Gate (Open collector) | 7400/74132/4011/4093/40107 |
| 7404 | Hex Inverter | 7405/7406/7414/4009/4049/4069 |
| 7405 | Hex Inverter (Open collector) | 7404/7406/7414/4009/4049/4069 |
| 7406 | Hex Inverter (Open collector) | 7404/7405/7414/4009/4049/4069 |
| 7407 | Hex Buffer | 74240/74241/74244/74365/74541/4050/4503/40106 |
| 7408 | Quad 2-Input AND Gate | 4081 |
| 7410 | Triple 3-Input NAND Gate | 4023 |
| 7411 | Triple 3-Input AND Gate | 4073 |
| 7414 | Hex Inverter (Schmitt trigger) | 7404/7405/7406/4009/4049/4069 |
| 7420 | Dual 4-Input NAND Gate | 4012 |
| 7421 | Dual 4-Input AND Gate | 4082 |
| 7427 | Triple 3-Input NOR Gate | 4025 |
| 7430 | 8-Input NAND Gate | 4068 |
| 7432 | Quad 2-Input OR Gate | 4071 |
| 7447 | BCD to 7-Segment decoder (15V) | 7448/4056/4511 |
| 7448 | BCD to 7-Segment decoder | 7447/4056/4511 |
| 7473 | Dual J-K Flip-Flop | 7476/4027 |
| 7474 | Dual D-Type Flip-Flop | 74175/74273/4013/40174 |
| 7476 | Dual J-K Flip-Flop | 7473/4027 |
| 7483 | 4-Bit Binary Full Adder | 74283 |
| 7485 | 4-Bit Magnitude Comparator | 4063/4585 |
| 7486 | Quad 2-Input XOR Gate | 74266/4030/4070/4077 |
| 7490 | Decade Counter | 74390/4017/4510 |
| 7493 | 4-Bit Binary Counter | 74161/74163/74193/74393/4029 |
| 74121 | Monostable Multivibrator | 74221/4098/4528 |
| 74132 | Quad 2-Input NAND Gate (Schmitt trigger) | 7400/7403/4011/4093/40107 |

| | | |
|---|---|---|
| 74137 | 3 to 8 Line Decoder | 74138 |
| 74138 | 3 to 8 Line Decoder | 74137 |
| 74154 | 4 to 16 Line Decoder | 4514 |
| 74161 | 4-Bit Binary Counter | 7493/74163/74193/74393/4029 |
| 74163 | 4-Bit Binary Counter | 7493/74161/74193/74393/4029 |
| 74161 | 8-Bit Shift Register | 74164/4014/4015 |
| 74164 | 8-Bit Shift Register | 74161/4014/4015 |
| 74175 | Quad D-Type Flip-Flop | 7474/74273/4013/40174 |
| 74193 | Dual 4-Bit Binary Counter | 7493/74161/74163/74393/4029 |
| 74221 | Dual Monostable Multivibrator | 74121/4098/4528 |
| 74240 | Octal Buffer | 7407/74241/74244/74365/74541/4050/ 4503/40106 |
| 74241 | Octal Buffer | 7407/74240/74244/74365/74541/4050/ 4503/40106 |
| 74244 | Octal Buffer | 7407/74240/74241/74365/74541/4050/ 4503/40106 |
| 74259 | 8-Bit Addressable Latch | 4099 |
| 74266 | Quad 2-Input XOR Gate | 7486/4030/4070/4077 |
| 74273 | Octal D-Type Flip-Flop | 7474/74175/4013/40174 |
| 74283 | 4-Bit Binary Full Adder | 7483 |
| 74365 | Hex Buffer | 7407/74240/74241/74244/74541/4050/ 4503/40106 |
| 74373 | Octal D-Type Latch (Inverting/ Tri-State) | 74374/74533/74573/74574/4042 |
| 74374 | Octal D-Type Latch | 74373/74533/74573/74574/4042 |
| 74390 | Dual Decade Counter | 7490/4017/4510 |
| 74393 | Dual 4-Bit Binary Counter | 7493/74161/74163/74193/4029 |
| 74533 | Octal D-Type Latch | 74373/74374/74573/74574/4042 |
| 74541 | Octal Buffer | 7407/74240/74241/74244/74365/4050/ 4503/40106 |
| 74573 | Octal D-Type Latch | 74373/74374/74533/74574/4042 |
| 74574 | Octal D-Type Latch | 74373/74374/74533/74573/4042 |

### CMOS Device Summary

| Device | Function | Similar Devices |
|---|---|---|
| 4001 | Quad 2-Input NOR Gate | 7402 |
| 4009 | Hex Inverter | 7404/7405/7406/7414/4049/4069 |
| 4011 | Quad 2-Input NAND Gate | 7400/7403/74132/4093/40107 |
| 4012 | Dual 4-Input NAND Gate | 7420 |

| | | |
|---|---|---|
| 4013 | Dual D-Type Flip-Flop | 7474/74175/74273/40174 |
| 4014 | 8-Bit Shift Register | 74161/74164/4015 |
| 4015 | 8-Bit Shift Register | 74161/74164/4014 |
| 4016 | 4-Channel Analogue Multiplexer | 4052 |
| 4017 | Decade Counter | 7490/74390/4510 |
| 4018 | BCD Counter (Presettable) | 4518 |
| 4021 | 8-Stage Shift Register | 4094 |
| 4023 | Triple 3-Input NAND Gate | 7410 |
| 4025 | Triple 3-Input NOR Gate | 7427 |
| 4026 | 7-Segment Display Decade Counter | 4033 |
| 4027 | Dual J-K Flip-Flop (Master/Slave ) | 7473/7476 |
| 4029 | 4-Bit Binary Counter | 7493/74161/74163/74193/74393 |
| 4030 | Quad 2-Input XOR Gate | 7486/74266/4070/4077 |
| 4033 | 7-Segment Display Decade Counter | 4026 |
| 4042 | Quad D-Type Latch (Clocked) | 74373/74374/74533/74573/74574 |
| 4047 | Multivibrator | 4538 |
| 4049 | Hex Inverter | 7404/7405/7406/7414/4009/4069 |
| 4050 | Hex Buffer | 7407/74240/74241/74244/74365/74541/4503/40106 |
| 4052 | Dual 4-Channel Analogue Multiplexer | 4016 |
| 4056 | BCD to 7-Segment Decoder | 7447/7448/4511 |
| 4063 | 4-Bit Magnitude Comparator | 7485/4585 |
| 4068 | 8-Input NAND Gate | 7430 |
| 4069 | Hex Inverter | 7404/7405/7406/7414/4009/4049 |
| 4070 | Quad 2-Input XOR Gate | 7486/74266/4030/4077 |
| 4071 | Quad 2-Input OR Gate | 7432 |
| 4073 | Triple 3-Input AND Gate | 7411 |
| 4077 | Quad 2-Input XOR Gate | 7486/74266/4030/4070 |
| 4081 | Quad 2-Input AND Gate | 7408 |
| 4082 | Dual 4-Input AND Gate | 7421 |
| 4093 | Quad 2-Input NAND Gate (Schmitt trigger) | 7400/7403/74132/4011/40107 |
| 4094 | 8-Stage Shift Register | 4021 |
| 4098 | Dual Monostable Multivibrator | 74121/74221/4528 |
| 4099 | 8-Bit Addressable Latch | 74259 |
| 4503 | Hex Buffer | 7407/74240/74241/74244/74365/74541/4050/40106 |
| 4510 | Decade Counter | 7490/74390/4017 |

| | | |
|---|---|---|
| 4511 | BCD to 7-Segment Decoder | 7447/7448/4056 |
| 4514 | 4 to 16 Line Decoder | 74154 |
| 4516 | Binary Counter (Presetable) | 4520 |
| 4518 | Dual BCD Counter | 4018 |
| 4520 | Dual Binary Counter | 4516 |
| 4528 | Dual Monostable Multivibrator | 74121/74221/4098 |
| 4538 | Multivibrator | 4047 |
| 4585 | 4-Bit Magnitude Comparator | 7485/4063 |
| 40106 | Hex Buffer (Schmitt trigger) | 7407/74240/74241/74244/74365/74541/4050/4503 |
| 40107 | Dual 2-Input NAND Gate | 7400/7403/74132/4011/4093 |
| 40174 | Hex D-Type Flip-Flop | 7474/74175/74273/4013 |

### NAND Gates

| Device | Function | Similar Devices |
|---|---|---|
| 7400 | Quad 2-Input NAND Gate | 7403/74132/4011/4093/40107 |
| 7403 | Quad 2-Input NAND Gate (Open collector) | 7400/74132/4011/4093/40107 |
| 7410 | Triple 3-Input NAND Gate | 4023 |
| 7420 | Dual 4-Input NAND Gate | 4012 |
| 7430 | 8-Input NAND Gate | 4068 |
| 74132 | Quad 2-Input NAND Gate (Schmitt trigger) | 7400/7403/4011/4093/40107 |
| 4011 | Quad 2-Input NAND Gate | 7400/7403/74132/4093/40107 |
| 4012 | Dual 4-Input NAND Gate | 7420 |
| 4023 | Triple 3-Input NAND Gate | 7410 |
| 4068 | 8-Input NAND Gate | 7430 |
| 4093 | Quad 2-Input NAND Gate (Schmitt trigger) | 7400/7403/74132/4011/40107 |
| 40107 | Dual 2-Input NAND Gate | 7400/7403/74132/4011/4093 |

### NOR Gates

| Device | Function | Similar Devices |
|---|---|---|
| 7402 | Quad 2-Input NOR Gate | 4001 |
| 7427 | Triple 3-Input NOR Gate | 4025 |
| 4001 | Quad 2-Input NOR Gate | 7402 |
| 4025 | Triple 3-Input NOR Gate | 7427 |

### AND Gates

| Device | Function | Similar Devices |
|---|---|---|
| 7408 | Quad 2-Input AND Gate | 4081 |
| 7411 | Triple 3-Input AND Gate | 4073 |
| 7421 | Dual 4-Input AND Gate | 4082 |
| 4073 | Triple 3-Input AND Gate | 7411 |
| 4081 | Quad 2-Input AND Gate | 7408 |
| 4082 | Dual 4-Input AND Gate | 7421 |

### OR Gates

| Device | Function | Similar Devices |
|---|---|---|
| 7432 | Quad 2-Input OR Gate | 4071 |
| 4071 | Quad 2-Input OR Gate | 7432 |

### XOR Gates

| Device | Function | Similar Devices |
|---|---|---|
| 7486 | Quad 2-Input XOR Gate | 74266/4030/4070/4077 |
| 74266 | Quad 2-Input XOR Gate | 7486/4030/4070/4077 |
| 4030 | Quad 2-Input XOR Gate | 7486/74266/4070/4077 |
| 4070 | Quad 2-Input XOR Gate | 7486/74266/4030/4077 |
| 4077 | Quad 2-Input XOR Gate | 7486/74266/4030/4070 |

### Buffers

| Device | Function | Similar Devices |
|---|---|---|
| 7407 | Hex Buffer | 74240/74241/74244/74365/74541/4050/4503/40106 |
| 74365 | Hex Buffer | 7407/74240/74241/74244/74541/4050/4503/40106 |
| 4050 | Hex Buffer | 7407/74240/74241/74244/74365/74541/4503/40106 |
| 4503 | Hex Buffer | 7407/74240/74241/74244/74365/74541/4050/40106 |
| 40106 | Hex Buffer (Schmitt trigger) | 7407/74240/74241/74244/74365/74541/4050/4503 |

### Inverters

| Device | Function | Similar Devices |
|---|---|---|
| 7404 | Hex Inverter | 7405/7406/7414/4009/4049/4069 |
| 7405 | Hex Inverter (Open collector) | 7404/7406/7414/4009/4049/4069 |
| 7406 | Hex Inverter (Open collector) | 7404/7405/7414/4009/4049/4069 |
| 7414 | Hex Inverter (Schmitt trigger) | 7404/7405/7406/4009/4049/4069 |
| 4009 | Hex Inverter | 7404/7405/7406/7414/4049/4069 |
| 4049 | Hex Inverter | 7404/7405/7406/7414/4009/4069 |
| 4069 | Hex Inverter | 7404/7405/7406/7414/4009/4049 |

### Interface Devices

| Device | Function | Similar Devices |
|---|---|---|
| 74240 | Octal Buffer | 7407/74241/74244/74365/74541/4050/4503/40106 |
| 74241 | Octal Buffer | 7407/74240/74244/74365/74541/4050/4503/40106 |
| 74244 | Octal Buffer | 7407/74240/74241/74365/74541/4050/4503/40106 |
| 74373 | Octal D-Type Latch (Inverting/Tri-State) | 74374/74533/74573/74574/4042 |
| 74374 | Octal D-Type Latch | 74373/74533/74573/74574/4042 |
| 74533 | Octal D-Type Latch | 74373/74374/74573/74574/4042 |
| 74541 | Octal Buffer | 7407/74240/74241/74244/74365 /4050/4503/40106 |
| 74573 | Octal D-Type Latch | 74373/74374/74533/74574/4042 |
| 74574 | Octal D-Type Latch | 74373/74374/74533/74573/4042 |

### Counters

| Device | Function | Similar Devices |
|---|---|---|
| 7490 | Decade Counter | 74390/4017/4510 |
| 7493 | 4-Bit Binary Counter | 74161/74163/74193/74393/4029 |
| 74161 | 4-Bit Binary Counter | 7493/74163/74193/74393/4029 |
| 74163 | 4-Bit Binary Counter | 7493/74161/74193/74393/4029 |
| 74193 | Dual 4-Bit Binary Counter | 7493/74161/74163/74393/4029 |
| 74390 | Dual Decade Counter | 7490/4017/4510 |
| 74393 | Dual 4-Bit Binary Counter | 7493/74161/74163/74193/4029 |
| 4017 | Decade Counter | 7490/74390/4510 |
| 4018 | BCD Counter (Presettable) | 4518 |
| 4029 | 4-Bit Binary Counter | 7493/74161/74163/74193/74393 |
| 4510 | Decade Counter | 7490/74390/4017 |
| 4516 | Binary Counter (Presetable) | 4520 |
| 4518 | Dual BCD Counter | 4018 |
| 4520 | Dual Binary Counter | 4516 |

### Latches and Flip-Flops

| Device | Function | Similar Devices |
|---|---|---|
| 7473 | Dual J-K Flip-Flop | 7476/4027 |
| 7474 | Dual D-Type Flip-Flop | 74175/74273/4013/40174 |
| 7476 | Dual J-K Flip-Flop | 7473/4027 |
| 74175 | Quad D-Type Flip-Flop | 7474/74273/4013/40174 |
| 74273 | Octal D-Type Flip-Flop | 7474/74175/4013/40174 |
| 74373 | Octal D-Type Latch (Inverting/Tri-State) | 74374/74533/74573/74574/4042 |
| 74374 | Octal D-Type Latch | 74373/74533/74573/74574/4042 |
| 74533 | Octal D-Type Latch | 74373/74374/74573/74574/4042 |
| 74573 | Octal D-Type Latch | 74373/74374/74533/74574/4042 |
| 74574 | Octal D-Type Latch | 74373/74374/74533/74573/4042 |
| 4013 | Dual D-Type Flip-Flop | 7474/74175/74273/40174 |
| 4027 | Dual J-K Flip-Flop (Master/Slave) | 7473/7476 |
| 4042 | Quad D-Type Latch (Clocked) | 74373/74374/74533/74573/74574 |
| 40174 | Hex D-Type Flip-Flop | 7474/74175/74273/4013 |

### Decoders

| Device | Function | Similar Devices |
|---|---|---|
| 7447 | BCD to 7-Segment decoder (15 V) | 7448/4056/4511 |
| 7448 | BCD to 7-Segment decoder | 7447/4056/4511 |
| 74137 | 3 to 8 Line Decoder | 74138 |
| 74138 | 3 to 8 Line Decoder | 74137 |
| 74154 | 4 to 16 Line Decoder | 4514 |
| 74259 | 8-Bit Addressable Latch | 4099 |
| 4026 | 7-Segment Display Decade Counter | 4033 |
| 4033 | 7-Segment Display Decade Counter | 4026 |
| 4056 | BCD to 7-Segment Decoder | 7447/7448/4511 |
| 4099 | 8-Bit Addressable Latch | 74259 |
| 4511 | BCD to 7-Segment Decoder | 7447/7448/4056 |
| 4514 | 4 to 16 Line Decoder | 74154 |

### Shift Registers

| Device | Function | Similar Devices |
|---|---|---|
| 74161 | 8-Bit Shift Register | 74164/4014/4015 |
| 74164 | 8-Bit Shift Register | 74161/4014/4015 |
| 4014 | 8-Bit Shift Register | 74161/74164/4015 |
| 4015 | 8-Bit Shift Register | 74161/74164/4014 |
| 4021 | 8-Stage Shift Register | 4094 |
| 4094 | 8-Stage Shift Register | 4021 |

## Computational Devices

| Device | Function | Similar Devices |
|---|---|---|
| 7483 | 4-Bit Binary Full Adder | 74283 |
| 7485 | 4-Bit Magnitude Comparator | 4063/4585 |
| 74283 | 4-Bit Binary Full Adder | 7483 |
| 4063 | 4-Bit Magnitude Comparator | 7485/4585 |
| 4585 | 4-Bit Magnitude Comparator | 7485/4063 |

## Monostables and Multivibrators

| Device | Function | Similar Devices |
|---|---|---|
| 74121 | Monostable Multivibrator | 74221/4098/4528 |
| 74221 | Dual Monostable Multivibrator | 74121/4098/4528 |
| 4047 | Multivibrator | 4538 |
| 4098 | Dual Monostable Multivibrator | 74121/74221/4528 |
| 4528 | Dual Monostable Multivibrator | 74121/74221/4098 |
| 4538 | Multivibrator | 4047 |

## Comprehensive Cross-Reference of 74xx and 155xx ICs

| IC | Pins | #/IC | Function | |
|---|---|---|---|---|
| 7400 | 14 | 4 | Dual Input NAND | К155ЛА3 |
| 7401 | 14 | 4 | Dual Input NAND gate (Open Collector) | К155ЛА8 |
| 7402 | 14 | 4 | Dual Input NOR gate | К155ЛЕ1 |
| 7403 | 14 | 4 | Dual Input NAND gate (OC) | К155ЛА9 |
| 7404 | 14 | 6 | Inverter | К155ЛН1 |
| 7405 | 14 | 6 | Inverter (OC) | К155ЛН2 |
| 7406 | 14 | 6 | 30 V/40 mA Inverter | К155ЛН3 |
| 7407 | 14 | 6 | Buffer/Driver (OC) | К155ЛП9 |
| 7408 | 14 | 4 | Dual Input AND gate | К155ЛИ1 |
| 7409 | 14 | 4 | Dual Input AND gate (OC) | К155ЛИ2 |
| 7410 | 14 | 4 | Triple Input NAND gate | К155ЛА4 |
| 7411 | 14 | 3 | Triple Input AND gate | К155ЛИ3 |
| 7412 | 14 | 3 | Triple Input AND gate | К155ЛА10 |
| 7413 | 14 | 2 | Schmitt Trigger NAND | К155ТЛ1 |
| 7414 | 14 | 6 | Schmitt Trigger Inverter | К155ТЛ2 |
| 7415 | 14 | 3 | Tri-State AND gate | К155ЛИ4 |
| 7416 | 14 | 6 | 15 V/40 mA Inverter | К155ЛН5 |

| 7417 | 14 | 6 | 15 V/40 mA Buffer | К155ЛП4 |
|---|---|---|---|---|
| 7420 | 14 | 2 | Quad Input NAND gate | К155ЛА1 |
| 7421 | 14 | 2 | Quad Input AND gate | К155ЛИ6 |
| 7422 | 14 | 2 | Quad Input NAND gate (OC) | К155ЛА7 |
| 7423 | 14 | 2 | Quad Input NOR gate with strobe | К155ЛЕ2 |
| 7424 | 14 | 2 | Quad Input NOR gate | |
| 7425 | 14 | 2 | Quad Input NOR gate with strobe | К155ЛЕ3 |
| 7426 | 14 | 4 | Dual Input NAND gate | К155ЛА11 |
| 7427 | 14 | 3 | Triple Input NOR gate | К155ЛЕ4 |
| 7428 | 14 | 4 | Dual Input NOR Buffer | К155ЛЕ5 |
| 7430 | 14 | 1 | 8 Input NAND gate | К155ЛА2 |
| 7432 | 14 | 4 | Dual Input OR gate | К155ЛЛ1 |
| 7433 | 14 | 4 | Dual Input NOR Buffer | К155ЛЕ11 |
| 7437 | 14 | 4 | Dual Input NAND Buffer | К155ЛА12 |
| 7438 | 14 | 4 | Dual Input NAND Buffer (OC) | К155ЛА13 |
| 7439 | 14 | 4 | Dual Input NAND Buffer (OC) | |
| 7440 | 14 | 2 | Quad Input NAND Buffer | К155ЛА6 |
| 7442 | 16 | • | BCD to Decimal Decoder/Driver | К155ИД6 |
| 7445 | 16 | • | BCD to Decimal Decoder/Driver | К155ИД24 |
| 7446 | 16 | • | BCD to 7 Segment LED Decoder/Driver | |
| 7447 | 16 | 1 | BCD to 7 Segment LED Decoder/Driver | |
| 7448 | 16 | • | BCD to 7 Segment LED Decoder | |
| 7449 | 14 | • | BCD to 7 Segment LED Decoder | К155ПП4 |
| 7450 | 14 | 2 | Dual Input AND inverter gate | К155ЛР1 |
| 7451 | 14 | 2 | Dual Input AND/OR | К155ЛР11 |
| 7453 | 14 | • | 4 Wide AND/OR inverter gate | К155ЛР3 |
| 7454 | 14 | 4 | Dual Input Inverter | К155ЛР13 |
| 7455 | • | • | 2-Wide 4-Input AND-OR-INVERT Gate (74H version is expandable) | К155ЛР4 |
| 7460 | • | • | Dual 4-input Expander | К155ЛД1 |
| 7464 | • | • | 4-2-3-2 Input AND/OR inverter gate | К155ЛР9 |
| 7465 | • | • | 4-2-3-2 Input AND/OR inverter gate | К155ЛР10 |
| 7470 | 14 | • | Gated edge-triggered J-K Flip-Flop | |
| 7472 | 14 | • | J-K Master/Slave Pulse-Triggered J-K FF | К155ТВ1 |
| 7473 | 14 | 2 | J-K Flip-Flop with Clear | |
| 7474 | 14 | 2 | D Flip-Flop With Preset and Clear | К155ТМ2 |
| 7475 | 16 | 1 | 4-bit Bistable Latches | К155ТМ7 |
| 7476 | 16 | 2 | J-K Flip-Flop | К155ТК3 |
| 7477 | 16 | 2 | 4-bit Bistable Latches | К155ТМ5 |

| | | | | |
|---|---|---|---|---|
| 7478 | 14 | 2 | Edge Triggered J-K Flip-Flop | К155ТВ14 |
| 7480 | 14 | • | Gated Full Adder | К155ИМ1 |
| 7481 | • | • | 16-bit Random Access Memory | К155РУ1 |
| 7482 | 16 | 1 | Single 2-bit Binary Full Adder | К155ИМ2 |
| 7483 | 16 | 1 | 4-bit Binary Full Adder | К155ИМ3 |
| 7484 | • | • | 16-bit Random Access Memory | К155РУ3 |
| 7485 | 16 | • | 4-bit Magnitude Comparator | К155СП1 |
| 7486 | 14 | 4 | Dual Input XOR gate | К155ЛП5 |
| 7489 | 16 | 1 | 64-bit RAM, open collector output | К155РУ2 |
| 7490 | 14 | • | Decade Counter | К155ИЕ2 |
| 7491 | 14 | • | 8-bit Shift Register | К155ИР2 |
| 7492 | • | • | Divide by 12 Counter | К155ИЕ4 |
| 7493 | 14 | • | Divide by 16 Counter | К155ИЕ5 |
| 7495 | 14 | • | 4-bit Right/Left Shift Register | К155ИР1 |
| 7496 | • | • | 5-bit Shift Register | |
| 7497 | 16 | • | 6-bit Rate Multiplier | К155ИЕ8 |
| 7498 | • | • | 4-bit Data Selector/Storage Register | К155ИР5 |
| 74100 | • | • | 8-bit Bistable Latch | К155ТК7 |
| 74107 | 14 | 2 | J-K Flip-Flop | К155ТВ6 |
| 74109 | 16 | 2 | J-K Pos. Flip-Flop | К155ТВ15 |
| 74112 | 16 | 2 | J-K Neg. Flip-Flop | К155ТВ9 |
| 74113 | 14 | 2 | J-K edge Triggered Flip-Flop w/ Preset | К155ТВ10 |
| 74114 | • | 2 | J-K Neg. Edge Flip-Flop w/Com Clk & Clr | К155ТВ11 |
| 74116 | 24 | 2 | 4-bit with clear latch | |
| 74120 | 16 | 2 | Pulse Sync/Driver | |
| 74121 | 14 | • | Monostable Multivibrator | К155АГ1 |
| 74122 | • | • | Retriggerable Monostable Multivibrator | |
| 74123 | 16 | • | Dual Reset Multivibrator | К155АГ3 |
| 74125 | 14 | • | Quad Bus Buffer gate | К155ЛП8 |
| 74126 | 14 | • | Quad Bus Buffer gate | К155ЛП14 |
| 74128 | 14 | 4 | Dual Input Schmitt Trigger NOR Buffer | К155ЛЕ6 |
| 74132 | 14 | 4 | Dual Input Schmitt Trigger NAND gate | К155ТЛ3 |
| 74133 | 16 | 1 | 13 Input NAND gate | |
| 74134 | 16 | 1 | 12 Input NAND gate Tri-State | К155ЛА19 |
| 74135 | 16 | 4 | XOR/NOR gate | |
| 74136 | • | 4 | XOR gate | К155ЛП12 |
| 74137 | 16 | • | 3 to 8 line Decoder/Dmultiplexer | |
| 74138 | 16 | • | Expandable 3/8 Decoder | К155ИД7 |
| 74139 | 16 | 2 | Expandable 2/4 Decoder | К155ИД14 |

| | | | | |
|---|---|---|---|---|
| 74140 | 14 | 2 | Quad Input NAND Line Driver | К155ЛА16 |
| 74141 | • | • | 1 of 10 Decoder/Driver/Nixie | К155ИД1 |
| 74143 | 24 | • | 4-bit 7-segment counter/latch | |
| 74145 | 16 | • | 10-4 Line Encoder | К155ИД10 |
| 74148 | 16 | • | Octal Encoder | К155ИВ1 |
| 74150 | 24 | • | 16 Input Multiplexer | К155КП1 |
| 74151 | 16 | • | 8 Input Multiplexer | К155КП7 |
| 74152 | 16 | • | 8 Line to 1-line Data Selector/Multiplexer | К155КП5 |
| 74153 | 16 | 2 | Quad Input Multiplexer | К155КП2 |
| 74154 | 24 | • | 1 of 16 Decoder/Demultiplexer | К155ИД3 |
| 74155 | 16 | 2 | 2-4 Decoder/Demultiplexer | К155ИД4 |
| 74156 | 16 | 2 | 2-4 Decoder/Demultiplexer | К155ИД5 |
| 74157 | 16 | 4 | Dual Input Multiplexer | К155КП16 |
| 74158 | 16 | 4 | 2/1 Multiplexer | К155КП18 |
| 74159 | 24 | • | 1 of 16 decoder/demultiplexer (OC) | К155ИД19 |
| 74160 | 16 | • | BCD Decade Counter | К155ИЕ9 |
| 74161 | 16 | • | Synchronous Binary Counter 4-bit | К155ИЕ10 |
| 74162 | 16 | • | BCD Decade Counter | К155ИЕ11 |
| 74163 | 16 | • | Synchronous Binary Counter | К155ИЕ18 |
| 74164 | 14 | • | Serial In/Parallel Out Register | К155ИР8 |
| 74165 | • | • | 8-bit Parallel to Serial Converter | К155ИР9 |
| 74166 | 16 | • | 8-bit Shift Register | К155ИР10 |
| 74167 | 16 | • | Sync Decade Rate Multiplier | |
| 74168 | 16 | • | 4-bit Up/Down Sync Decade Counter | К155ИЕ16 |
| 74169 | 16 | • | 4-bit Up/Down Sync Binary Counter | К155ИЕ17 |
| 74170 | 16 | • | 4x4 Register File | К155ИР32 |
| 74172 | 24 | • | 16-bit Mult Port Register File | К155РП3 |
| 74173 | 16 | • | 4-bit D Register 3-State | К155ИР15 |
| 74174 | 16 | 6 | D Flip-Flop | К155ТМ9 |
| 74175 | 16 | 4 | D Flip-Flop | К155ТМ8 |
| 74176 | 14 | • | Preset Decade Counter 35 MHz | |
| 74177 | 14 | • | Preset Binary Counter 35 MHz | |
| 74180 | 14 | • | 9-bit Odd/Even Parity Generator | К155ИП2 |
| 74181 | 24 | • | 4-bit Arithmetic Logic | К155ИП3 |
| 74182 | 16 | • | Look ahead Carry Generator | К155ИП4 |
| 74183 | • | • | Dual Carry-Save Full Adder | К155ИМ5 |
| 74184 | • | • | BCD to Binary Converter | К155ПР6 |
| 74185 | • | • | Binary to BCD Converter | К155ПР7 |
| 74188 | 16 | • | 256-bit PROM | |

| | | | | |
|---|---|---|---|---|
| 74189 | 16 | • | 16x4 SRAM | К155РУ8 |
| 74190 | 16 | • | BCD/Decade Up/Down Counter | К155ИЕ12 |
| 74191 | 16 | • | Up/Down Binary Converter 4-bit | К155ИЕ13 |
| 74192 | 16 | • | Up/Down Decade Counter BCD | К155ИЕ6 |
| 74193 | 16 | • | Up/Down Binary Counter 4-bit | К155ИЕ7 |
| 74194 | 16 | • | 4-bit Bidirectional Universal Shift Register | К155ИР11 |
| 74195 | 16 | • | Universal 4-bit Shift Register | К155ИР12 |
| 74196 | 16 | • | Preset Decade Counter | К155ИЕ14 |
| 74197 | 14 | • | Preset Binary Counter | К155ИЕ15 |
| 74198 | 24 | • | 8-bit Bidirectional Universal Shift Reg | К155ИР13 |
| 74219 | • | • | 64-Bit RAM w/ Tristate Outputs | |
| 74221 | 16 | 2 | Monostable Multivibrator | К155АГ4 |
| 74238 | 16 | • | 3 to 8 line Decoder/Demultiplexer | К155ИД19 |
| 74240 | 20 | • | Tri-State Octal Line Driver | К155АП3 |
| 74241 | 20 | • | Tri-State Octal Line Driver | К155АП4 |
| 74242 | • | • | Quad Bus Transceiver | К155ИП6 |
| 74243 | 14 | • | Tri-State Quad Bus Transceiver | К155ИП7 |
| 74244 | 20 | 8 | Tri-State Octal Line Driver | К155АП5 |
| 74245 | 20 | • | Tri-State Octal Transcender | К155АП6 |
| 74247 | 16 | • | BCD to 7-Segment Decoder/Driver | К155ИД18 |
| 74248 | 16 | • | BCD to 7-Segment Decoder/Driver | |
| 74249 | 16 | • | BCD to 7-Segment Decoder/Driver | |
| 74251 | 16 | • | Tri-State 8/1 Multiplexer | К155КП15 |
| 74253 | 16 | 2 | Tri-State 4/1 Multiplexer | К155КП12 |
| 74257 | 16 | 4 | 2/1 Multiplexer | К155КП11 |
| 74258 | 16 | 4 | 2/1 Multiplexer | К155КП14 |
| 74259 | 16 | • | 8-bit Addressable Latch | К155ИР30 |
| 74260 | 14 | 2 | 5 Input NOR gate | К155ЛЕ7 |
| 74265 | 16 | 4 | Comp-Out Element | |
| 74266 | 14 | 4 | Dual Input XNOR gate (OC) | |
| 74273 | 20 | 8 | D Flip-Flop | К155ИР35 |
| 74278 | 14 | • | 4-bit Priority Register | |
| 74279 | 16 | • | Quad Set/Reset Latch | К155ТР2 |
| 74280 | 14 | • | Octal D Flip-Flop | К155ИП5 |
| 74283 | 16 | • | 4-bit Full Adder | К155ИМ8 |
| 74287 | 16 | • | 1024-bit PROM | |
| 74288 | 16 | • | 256-bit PROM | |
| 74289 | 16 | • | 64-bit RAM Open Collector Outputs | К155РУ9 |
| 74290 | 14 | • | Decade Counter | |

| | | | | |
|---|---|---|---|---|
| 74293 | 14 | • | 4-bit Binary Ripple Counter | |
| 74298 | 16 | 4 | Two Port Register | К155КП13 |
| 74299 | 20 | • | 8 Input Shift/Storage Register | К155ИР24 |
| 74322 | • | 8 | Serial/Parallel Register with Sign Extend | К155ИР28 |
| 74323 | • | 8 | Shift/Storage Reg w/Sync Rst & Common I/O | К155ИР29 |
| 74348 | 16 | • | 8 to 3 Priority Register | К155ИВ2 |
| 74350 | • | • | 4-bit Shifter with 3-state Outputs | К155ИР42 |
| 74352 | • | 2 | 4 input Multiplexer (MUX) | К155КП19 |
| 74353 | 16 | 2 | 4-bit Multiplexer, Inverting Output | К155КП17 |
| 74365 | 16 | 6 | Three State Buffer | К155ЛП10 |
| 74366 | 16 | 6 | Three State Inverting Buffer | К155ЛН6 |
| 74367 | 16 | 6 | Three State Buffer | К155ЛП11 |
| 74368 | 16 | 6 | Three State Inverting Buffer | К155ЛН7 |
| 74373 | 20 | • | Octal Transparent (D) Latch | К155ИР22 |
| 74374 | 20 | 8 | D Flip-Flop Tri-State | К155ИР23 |
| 74375 | 16 | • | 4-bit Bisettable Latch | |
| 74377 | 20 | 8 | D Flip-Flop With Clock Enable | К155ИР27 |
| 74378 | 16 | 6 | D Flip-Flop With Clock Enable | |
| 74379 | 16 | 4 | D Flip-Flop With Clock Enable | К155ТМ10 |
| 74381 | 20 | • | ALU/Function Generator | К155ИК2 |
| 74382 | • | • | 4-bit Arithmetic Logic Unit (ALU) | |
| 74385 | • | • | Quad 4-bit Adder/Subtractor | К155ИМ7 |
| 74386 | 14 | 4 | Dual Input XOR gate | |
| 74387 | 16 | • | 1024-bit PROM (OC) | |
| 74390 | 16 | 2 | Decade Ripple Counter | К155ИЕ20 |
| 74393 | 14 | 2 | Modulo 16 Counter | К155ИЕ19 |
| 74395 | • | • | 4-bit Universal Shift Registers with Three-State Outputs | К155ИР25 |
| 74396 | 16 | 8 | Storage Register | К155ИР43 |
| 74398 | • | 4 | 2-port Register | |
| 74399 | 16 | 4 | Dual Input Multiplexer, Dual Port Register | К155КП20 |
| 74401 | • | • | Cyclic Redundancy Check (CRC) Gen/Check | |
| 74402 | • | • | Serial Data Polynomial Generator/Checker | |
| 74413 | • | • | 64x4 FIFO w/Parallel I/O | |
| 74425 | 14 | • | Quad Bus Buffer | |
| 74426 | 14 | • | Quad Bus Buffer | |
| 74433 | • | • | FIFO | |

| | | | | |
|---|---|---|---|---|
| 74450 | • | • | 16-to-1 Multiplexer with Complementary Outputs | К155ЛР7 |
| 74451 | • | • | Dual 8-to-1 Multiplexer | К155ЛИ5 |
| 74452 | • | • | Dual Decade Counter, Synchronous | К155ЛА18 |
| 74465 | • | • | Octal Buffer with Three-State Outputs | К155АП14 |
| 74472 | 20 | • | 512x8 3-state PROM | |
| 74474 | 24 | • | 512x8 3-state PROM | |
| 74490 | 16 | 2 | BCD decade ripple counter | |
| 74521 | 20 | • | Octal Comparator | |
| 74524 | • | • | Octal Registered Comparator | |
| 74533 | • | • | Octal Transparent Latch w/tristate output | К155ИР40 |
| 74534 | 8 | | D Flip-Flop | К155ИР41 |
| 74537 | • | • | 1 of 10 decoder w/tristate outputs | К155ИД22 |
| 74538 | • | • | 1 of 8 decoder w/tristate outputs | |
| 74539 | • | 2 | 1 of 4 decoder w/tristate outputs | |
| 74540 | 20 | 8 | Tri-State Inverter Buffer | К155АП12 |
| 74541 | 20 | 8 | Non-Inverting Buffer/Line Driver | К155АП13 |
| 74543 | • | 8 | Registered Transceiver | |
| 74544 | • | 8 | Registered Transceiver (Inverting) | |
| 74545 | • | 8 | Bidirectional Transceiver w/tristate outputs | |
| 74552 | • | 8 | Registered Transceiver w/parity & flags | |
| 74563 | • | 8 | D Latch w/tristate outputs | |
| 74564 | • | 8 | D Flip-Flop w/tristate outputs | |
| 74569 | • | • | 4-bit bidirectional counter w/tristate out | |
| 74571 | 16 | • | 512x4 3-State PROM | |
| 74573 | 20 | 8 | D Latch w/tristate outputs | К155ИР33 |
| 74574 | 20 | 8 | D Flip-Flop w/tristate outputs | К155ИР37 |
| 74579 | • | • | 8-bit binary counter w/tristate outputs | |
| 74583 | • | • | 4-bit BCD Adder | |
| 74590 | 16 | 1 | 8-bit Three-State Binary Counter | |
| 74593 | • | • | 8-Bit Binary Counter with Input Registers and Three-State Outputs | К155ИЕ21 |
| 74595 | 16 | • | 8-bit Serial to Parallel Shift Register | |
| 74620 | • | 8 | Inverting bus transceiver w/tristate outputs | К155АП25 |
| 74623 | • | 8 | Inverting bus transceiver w/tristate outputs | К155АП26 |
| 74624 | 14 | 1 | Voltage Controlled Oscillator | К155ГГ6 |
| 74626 | • | • | Dual Voltage-Controlled Oscillator with Enable Control, Two-Phase Outputs | К155ГГ2 |
| 74629 | 16 | 2 | Voltage Controlled Oscillator | |
| 74640 | 20 | • | Tri-State Octal Bus Transceiver | К155АП9 |

| | | | | |
|---|---|---|---|---|
| 74641 | • | • | Octal Bus Transceiver (OC) | К155АП7 |
| 74643 | • | • | Octal Bus Transceiver with Mix of Inverting and Noninverting Three-State Outputs | К155АП16 |
| 74644 | • | • | Octal Bus Transceiver (OC) | |
| 74645 | 20 | • | Tri-State Octal Bus Transceiver | К155АП8 |
| 74646 | 24 | 8 | Trans/Reg Three-State (.3" Wide) | К155ВА1 |
| 74648 | • | 8 | Bus transceiver w/tristate output | К155ВА2 |
| 74651 | • | • | transceiver/register | К155АП17 |
| 74652 | • | • | transceiver/register | К155АП24 |
| 74657 | • | 8 | bidir xceive w/8-bit parity gen/chk & 3-state | |
| 74670 | 16 | • | 4x4 Register File 3-State | К155ИР26 |
| 74673 | • | • | 16-bit Ser/Parallel shift reg w/com ser i/o | |
| 74675 | • | • | 16-bit Serial/Parallel shift register | |
| 74676 | • | • | 16-bit Parallel/Serial shift register | |
| 74682 | 20 | • | 8-bit Comparator | |
| 74684 | 20 | • | 8-bit Comparator | |
| 74685 | 20 | • | 8-bit Comparator | |
| 74688 | 20 | • | 8-bit Magnitude Comparator | |
| 74779 | • | • | 8-bit bidir binary counter w/tristate outputs | |
| 74794 | • | • | 8-bit register w/readback | |
| 74804 | • | • | Hex 2-input NAND Drivers | К155ЛА20 |
| 74805 | • | • | Hex 2-input NOR Drivers | К155ЛЕ8 |
| 74808 | • | • | Hex 2-input AND Drivers | К155ЛИ7 |
| 74821 | • | • | 10-bit D Flip Flop | |
| 74823 | • | • | 10-bit D Flip Flop | |
| 74825 | • | • | 8-bit D Flip Flop | |
| 74827 | • | • | 10-bit buffer/line driver | |
| 74828 | • | • | 10-bit buffer/line driver | |
| 74832 | • | • | Hex 2-input OR Drivers | К155ЛЛ3 |
| 74841 | • | • | 10-bit transparent latch | |
| 74843 | • | • | 9-bit transparent latch | |
| 74845 | • | • | 8-bit transparent latch | |
| 74881 | • | • | 4-bit Arithmetic/Logic Unit/Function Generator | К155ИП14 |
| 74882 | • | • | 32-bit Lookahead Carry Generator | К155ИП16 |
| 74899 | • | • | 9-bit latchable transceiver w/parity gen/chk | |
| 74900 | • | • | 9-bit 3port latchable datapath multiplexer | |
| 74902 | 14 | 6 | Non-Inverting TTL Buffer | |
| 74903 | 14 | 6 | Inverting Buffer | |
| 74905 | 24 | • | 12-bit Successive Approximation Register | |

| | | | | |
|---|---|---|---|---|
| 74906 | 14 | 6 | Open Drain N-Channel Buffer | |
| 74907 | 14 | 6 | Open Drain P-Channel Buffer | |
| 74911 | 28 | • | 4-digit expandable Display Cont | |
| 74921 | 22 | • | 1024-bit Static SG CMOS RAM | |
| 74922 | 18 | • | 16-key Encoder | |
| 74923 | 20 | • | 20-key Encoder | |
| 74925 | 16 | 1 | 4-digit Counter/Multiplexer/7-Seg Display | |
| 74928 | 18 | 1 | 4-digit Counter/Multiplexer/7-Seg Display | |

# APPENDIX D

## 74 SERIES IC PIN ASSIGNMENTS

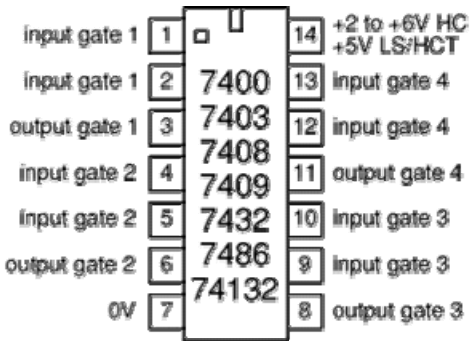
### Gates

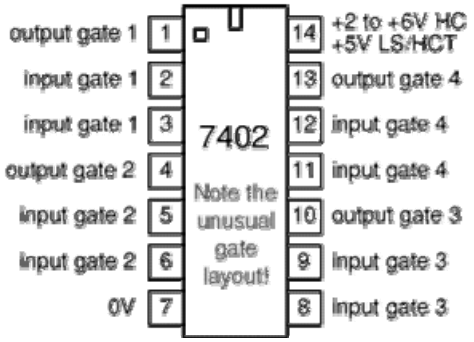
#### Quad 2-input gates

- 7400 quad 2-input NAND
- 7403 quad 2-input NAND with open collector outputs
- 7408 quad 2-input AND
- 7409 quad 2-input AND with open collector outputs
- 7432 quad 2-input OR
- 7486 quad 2-input EX-OR
- 74132 quad 2-input NAND with Schmitt trigger inputs

The 74132 has Schmitt trigger inputs to provide good noise immunity. They are ideal for slowly changing or noisy signals.

- 7402 quad 2-input NOR
  Note the unusual gate layout.

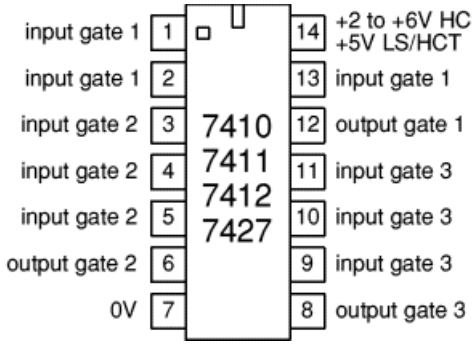



#### Triple 3-input gates

- 7410 triple 3-input NAND
- 7411 triple 3-input AND
- 7412 triple 3-input NAND with open collector outputs
- 7427 triple 3-input NOR

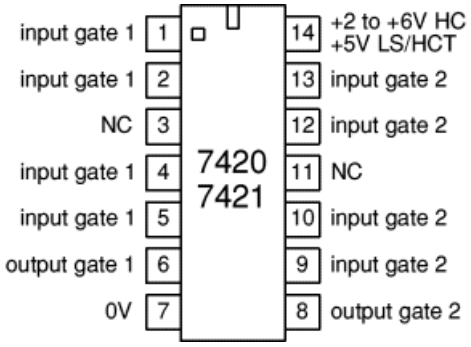
Notice how gate 1 is spread across the two sides of the package.



### Dual 4-input gates

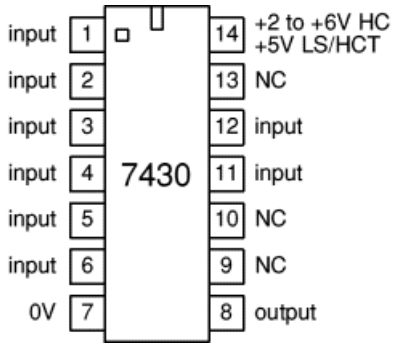
- 7420 dual 4-input NAND
- 7421 dual 4-input AND

NC = No Connection (a pin that is not used).




### 7430 8-input NAND gate

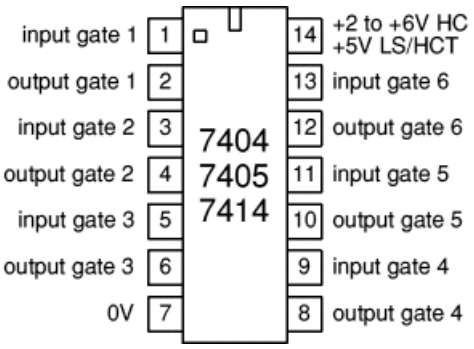
NC = No Connection (a pin that is not used).




### Hex NOT gates

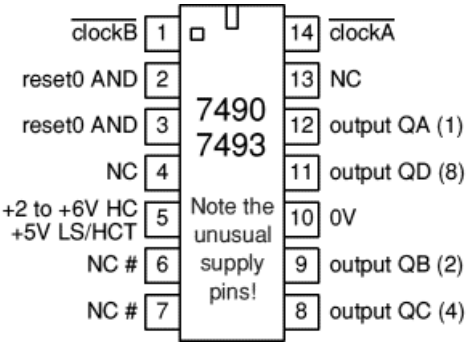
- 7404 hex NOT
- 7405 hex NOT with open collector outputs
- 7414 hex NOT with Schmitt trigger inputs

The 7414 has Schmitt trigger inputs to provide good noise immunity. They are ideal for slowly changing or noisy signals.



## Counters

### 7490 decade (0-9) ripple counter
### 7493 4-bit (0-15) ripple counter




***NC = No Connection (a pin that is not used). # on the 7490 pins 6 and 7 connect to an internal AND gate for resetting to 9.***

These are **ripple** counters so beware that glitches may occur in any logic gate systems connected to their outputs due to the slight delay before the later counter outputs respond to a clock pulse.

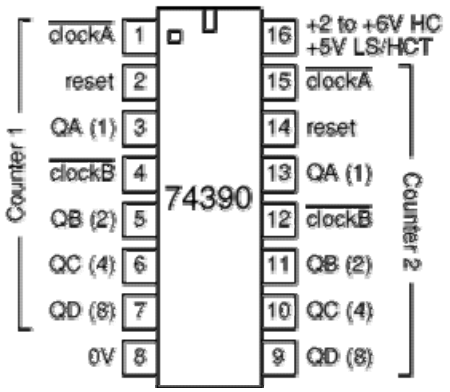
The count advances as the **clock** input becomes low (on the falling-edge), this is indicated by the bar over the clock label. This is the usual clock behaviour of ripple counters and it means a counter output can directly drive the clock input of the next counter in a chain.

**The counter is in two sections:** clock A-QA and clock B-QB-QC-QD. For normal use connect QA to clock B to link the two sections, and connect the external clock signal to clock A.

For normal operation at least one **reset0** input should be low, making both high resets the counter to zero (0000, QA-QD low). Note that the 7490 has a pair of **reset9** inputs on pins 6 and 7, these reset the counter to nine (1001) so at least one of them must be low for counting to occur.

**Counting to less than the maximum** (9 or 15) can be achieved by connecting the appropriate output(s) to the two **reset0** inputs. If only one reset input is required the two inputs can be connected together. For example: to count 0 to 8 connect QA (1) and QD (8) to the reset inputs.

### 74390 dual decade (0-9) ripple counter




The 74390 contains two separate decade (0 to 9) counters, one on each side of the chip. They are **ripple** counters so beware that glitches may occur in any logic gate systems connected to their outputs due to the slight delay before the later counter outputs respond to a clock pulse.

The count advances as the **clock** input becomes low (on the falling-edge), this is indicated by the bar over the clock label. This is the usual clock behaviour of ripple counters and it means a counter output can directly drive the clock input of the next counter in a chain.

**Each counter is in two sections:** clock A-QA and clock B-QB-QC-QD. For normal use connect QA to clock B to link the two sections, and connect the external clock signal to clock A.

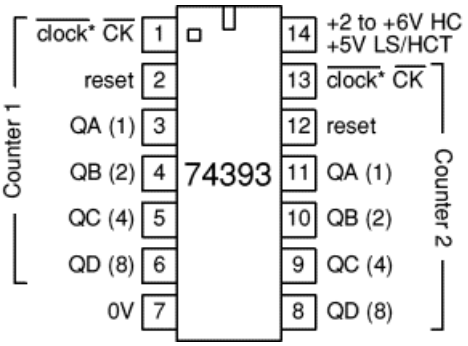
For normal operation the **reset** input should be low, making it high resets the counter to zero (0000, QA-QD low).

**Counting to less than 9** can be achieved by connecting the appropriate output(s) to the reset input, using an AND gate if necessary. For example: to count 0 to 7 connect QD (8) to reset, to count 0 to 8 connect QA (1) and QD (8) to reset using an AND gate.

**Connecting ripple counters in a chain:** please see 74393 below.

**74393 dual 4-bit**
**(0-15) ripple counter**

The 74393 contains two separate 4-bit (0 to 15) counters, one on each side of the chip. They are **ripple** counters so beware that glitches may occur in logic systems connected to their outputs due to the slight delay before the later outputs respond to a clock pulse.



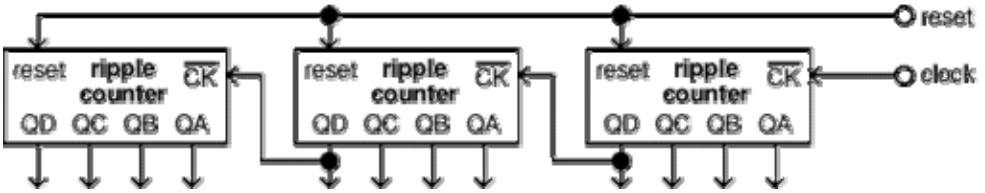

The count advances as the **clock** input becomes low (on the falling-edge), this is indicated by the bar over the clock label. This is the usual clock behaviour of ripple counters and it means a counter output can directly drive the clock input of the next counter in a chain.

For normal operation the **reset** input should be low, making it high resets the counter to zero (0000, QA-QD low).

**Counting to less than 15** can be achieved by connecting the appropriate output(s) to the reset input, using an AND gate if necessary. For example to count 0 to 8 connect QA (1) and QD (8) to reset using an AND gate.

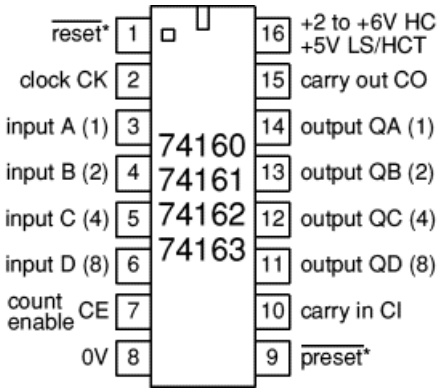
### Connecting ripple counters in a chain

The diagram below shows how to link ripple counters in a chain, notice how the highest output QD of each counter drives the **clock** input of the next counter.




### 74160-3 synchronous counters

- 74160 synchronous decade counter (standard reset)
- 74161 synchronous 4-bit counter (standard reset)
- 74162 synchronous decade counter (synchronous reset)
- 74163 synchronous 4-bit counter (synchronous reset)




**** reset and preset are both active-low***
***preset is also known as parallel enable (PE)***

These are **synchronous** counters so their outputs change precisely together on each clock pulse. This is helpful if you need to connect their outputs to logic gates because it avoids the glitches which occur with ripple counters.

The count advances as the **clock** input becomes high (on the rising-edge). The **decade** counters count from 0 to 9 (0000 to 1001 in binary). The **4-bit** counters count from 0 to 15 (0000 to 1111 in binary).

For normal operation (counting) the **reset**, **preset**, **count enable** and **carry in** inputs should all be high. When **count enable** is low the clock input is ignored and counting stops.

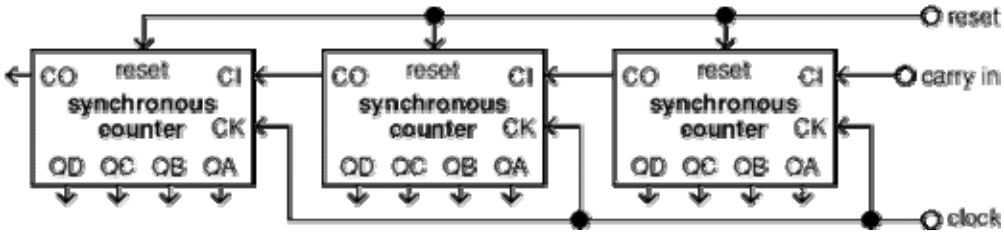
The counter may be **preset** by placing the desired binary number on the **inputs A-D**, making the **preset** input low, and applying a positive pulse to the **clock** input. The **inputs A-D** may be left unconnected if not required.

The **reset** input is active-low so it should be high (+Vs) for normal operation (counting). When low it resets the count to zero (0000, QA-QD low), this happens immediately with the 74160 and 74161 (**standard reset**), but with the 74162 and 74163 (**synchronous reset**) the reset occurs on the rising-edge of the clock input.

**Counting to less than the maximum** (15 or 9) can be achieved by connecting the appropriate output(s) through a NOT or NAND gate to the reset input. For the 74162 and 74163 (**synchronous reset**) you must use the output(s) representing **one less** than the reset count you require, e.g. to reset on 7 (counting 0 to 6) use QB (2) and QC (4).

### Connecting synchronous counters in a chain

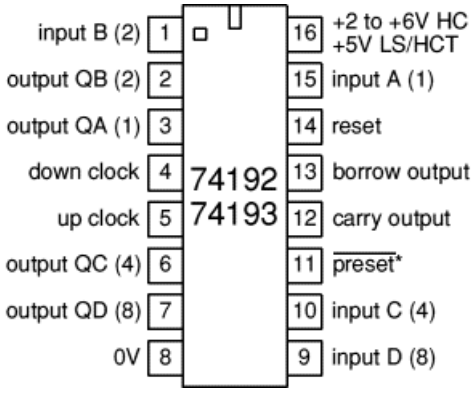
The diagram below shows how to link synchronous counters such as 74160-3, notice how all the **clock (CK)** inputs are linked. **Carry out (CO)** is used to feed the **carry in (CI)** of the next counter. **Carry in (CI)** of the first 74160-3 counter should be high.




### 74192 up/down decade (0-9) counter
### 74193 up/down 4-bit (0-15) counter

These are **synchronous** counters so their outputs change precisely together on each clock pulse. This is helpful if you need to connect their outputs to logic gates because it avoids the glitches which occur with ripple counters.

These counters have separate clock inputs for counting up and down. The count increases as the **up clock** input becomes high (on the rising-edge). The




** preset is active-low*

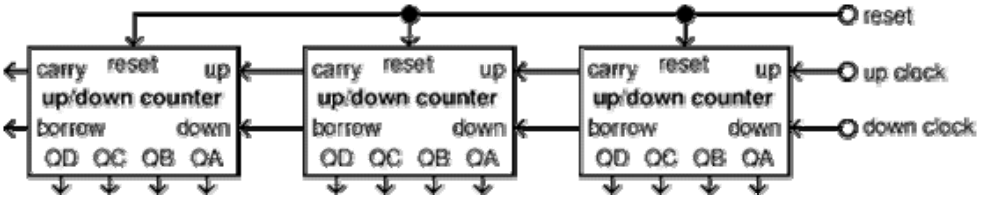
count decreases as the **down clock** input becomes high (on the rising-edge). In both cases the other clock input should be high.

For normal operation (counting) the **preset** input should be high and the **reset** input low. When the **reset** input is high it resets the count to zero (0000, QA-QD low)

The counter may be **preset** by placing the desired binary number on the **inputs A-D** and briefly making the **preset** input low. Note that a clock pulse is not required to preset, unlike the 74160-3 counters. The **inputs A-D** may be left unconnected if not required.

**Connecting counters with separate up and down clock inputs in a chain**

The diagram below shows how to link 74192-3 up/down counters with separate up and down clock inputs, notice how **carry** and **borrow** are connected to the **up clock** and **down clock** inputs respectively of the next counter.

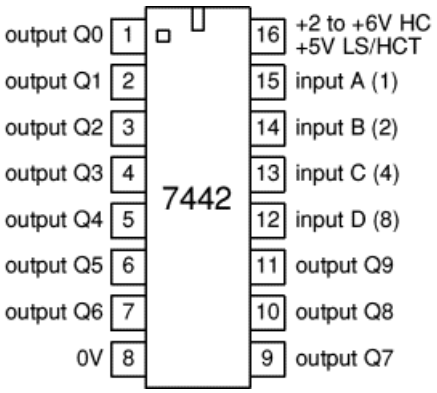



**Decoders**

**7442 BCD to decimal (1 of 10) decoder**

The 7442 **outputs** are **active-low** which means they become low when selected but are high at other times. They can sink up to about 20 mA.

The appropriate output becomes low in response to the BCD (binary coded decimal) input. For example an input of binary 0101 (=5) will make output Q5 low and all other outputs high.

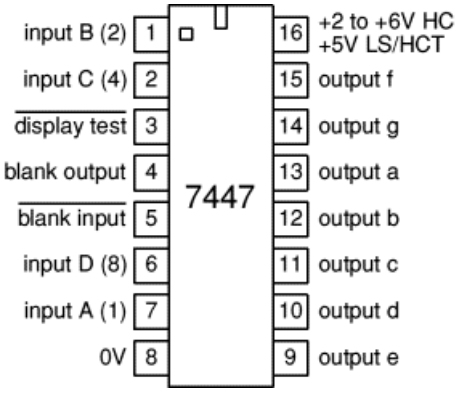



The 7442 is a BCD (binary coded decimal) decoder intended for input values 0 to 9 (0000 to 1001 in binary). With inputs from 10 to 15 (1010 to 1111 in binary) all outputs are high.

Note that the 7442 can be used as a **1-of-8 decoder** if input D is held low.

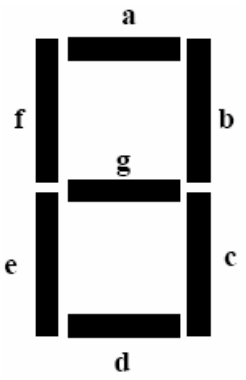
## 7-segment Display Drivers

### 7447 BCD to 7-segment display driver




The appropriate **outputs a-g** become low to display the BCD (binary coded decimal) number supplied on **inputs A-D**. The 7447 has **open collector** outputs a-g which can sink up to 40 mA. The 7-segment display segments must be connected between +Vs and the outputs with a resistor in series (330 Ω with a 5 V supply). A **common anode** display is required.

**Display test** and **blank input** are active-low so they should be high for normal operation. When **display test** is low all the display segments should light (showing number 8).




If the **blank input** is low the display will be blank when the count input is zero (0000). This can be used to blank leading zeros when there are several display digits driven by a chain of counters. To achieve this **blank output** should be connected to **blank input** of the next display down the chain (the next most significant digit).

The 7447 is intended for BCD (binary coded decimal) which is input values 0 to 9 (0000 to 1001 in binary). Inputs from 10 to 15 (1010 to 1111 in binary) will light odd display segments but will do no harm.

## Pin Assignments of 74 Series ICs

*From: http://www.fsref.com/Fatal/FE200202.SHTML*

### 7400 Quad Dual-Input NAND Gate


X = A NAND B

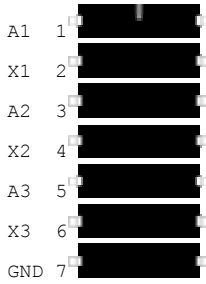
| NAND: | A | B | X |
|---|---|---|---|
| | 0 | 0 | 1 |
| | 0 | 1 | 1 |
| | 1 | 0 | 1 |
| | 1 | 1 | 0 |

### 7402 Quad Dual-Input NOR Gate


X = A NOR B

| NOR: | A | B | X |
|---|---|---|---|
| | 0 | 0 | 1 |
| | 0 | 1 | 0 |
| | 1 | 0 | 0 |
| | 1 | 1 | 0 |

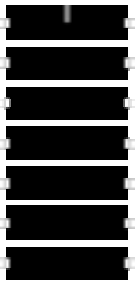
### 7404 Hex Inverter




X = -A

| INV: | A | X |
|---|---|---|
| | 0 | 1 |
| | 1 | 0 |

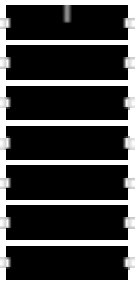
## 7408 Quad Dual-Input AND Gate




X = A AND B

| AND: A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

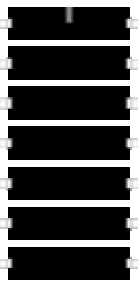
## 7410 Tri Triple-Input NAND Gate




X = A NAND B NAND C

| NAND: A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

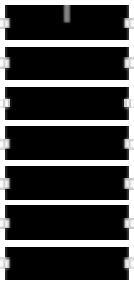
## 7411 Tri Triple-Input AND Gate




X = A AND B AND C

| AND: A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

## 7420 Dual Quad-Input NAND Gate



```
A1  1          14 Vcc   X = A NAND B NAND C NAND  DNAND:  A B C D X    A B C D X
B1  2          13 A2                                      0 0 0 0 1    1 0 0 0 1
NC  3          12 B2                                      0 0 0 1 1    1 0 0 1 1
C1  4          11 NC                                      0 0 1 0 1    1 0 1 0 1
D1  5          10 C2                                      0 0 1 1 1    1 0 1 1 1
X1  6          9  D2                                      0 1 0 0 1    1 1 0 0 1
GND 7          8  X2                                      0 1 0 1 1    1 1 0 1 1
                                                          0 1 1 0 1    1 1 1 0 1
                                                          0 1 1 1 1    1 1 1 1 0
```

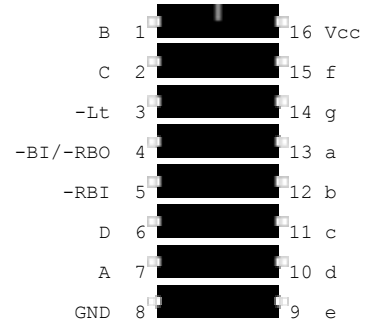
## 7427 Tri Triple-Input NOR Gate

```
B1  1          14 Vcc          X = A NOR B NOR C        NOR: A B C X
A1  2          13 C1                                         0 0 0 1
C2  3          12 X1                                         0 0 1 0
B2  4          11 C3                                         0 1 0 0
A2  5          10 B3                                         0 1 1 0
X2  6          9  A3                                         1 0 0 0
GND 7          8  X3                                         1 0 1 0
                                                             1 1 0 0
                                                             1 1 1 0
```

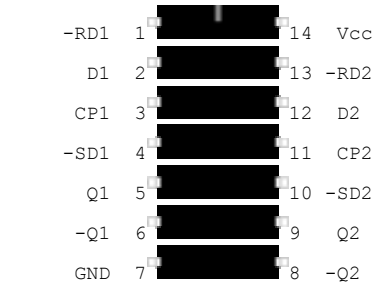
## 7432 Quad Dual-Input OR Gate

```
A1  1          14 Vcc          X = A OR B               OR:  A B X
B1  2          13 B4                                         0 0 0
X1  3          12 A4                                         0 1 1
A2  4          11 X4                                         1 0 1
B2  5          10 B3                                         1 1 1
X2  6          9  A3
GND 7          8  X3
```

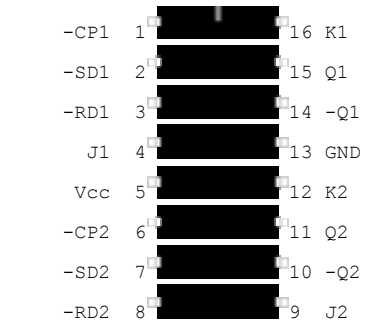
## 7447 BCD to 7-Segment LED Decoder
## 7448 BCD to 7-Segment Decoder



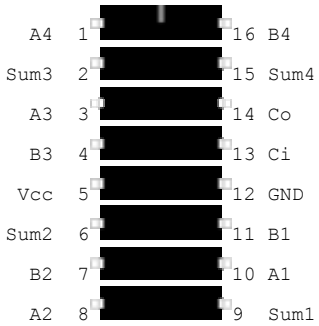

## 7474 Dual D-Type Flip-Flop



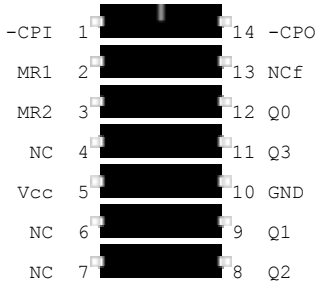

## 7476 Dual J-K Flip-Flop



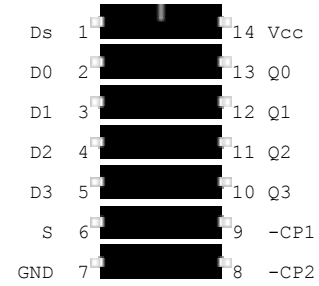
## 7483 4-Bit Full Adder With Fast Carry



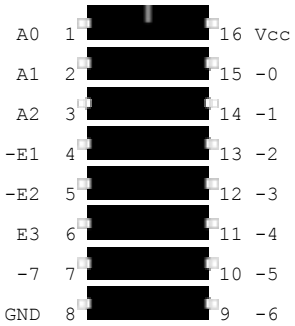

## 7493 4-Bit binary Ripple Counter



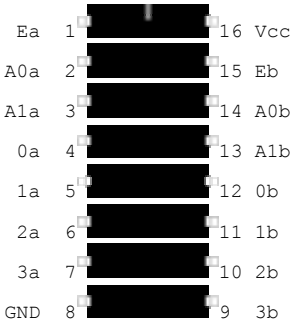

## 7495 4-Bit Shift Register



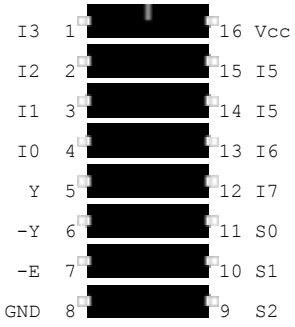
### 74138 3 Line to 8 Line Decoder



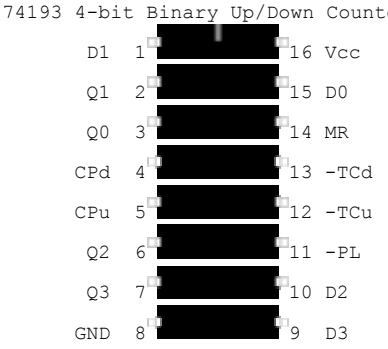

### 74139 Dual 4-Line Demultiplexer



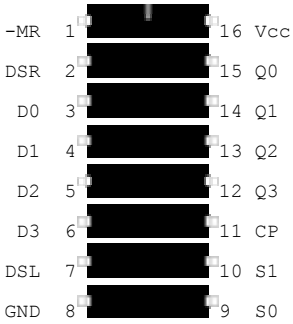

### 74151 Multiplexer



## 74192 BCD Decade Up/Down Counter




## 74194 4-Bit BiDirectional Shift Register



## APPENDIX E

# 14.2 mm (0.56 inch) Seven Segment Displays

## Technical Data

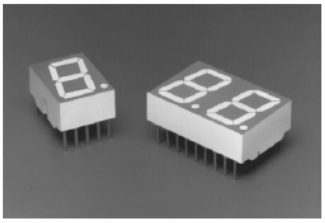
**HDSP-550x Series**
**HDSP-552x Series**
**HDSP-560x Series**
**HDSP-562x Series**
**HDSP-570x Series**
**HDSP-572x Series**
**HDSP-H15x Series**

### Features

- **Industry Standard Size**
- **Industry Standard Pinout**
  15.24 mm (0.6 in.) DIP Leads on 2.54 mm (0.1 in.) Centers
- **Choice of Colors**
  AlGaAs Red, High Efficiency Red, Yellow, Green
- **Excellent Appearance**
  Evenly Lighted Segments
  Mitered Corners on Segments
  Gray Package Gives Optimum Contrast
  ± 50° Viewing Angle
- **Design Flexibility**
  Common Anode or Common Cathode
  Single and Dual Digits
  Right Hand Decimal Point
  ± 1. Overflow Character
- **Categorized for Luminous Intensity**
  Yellow and Green Categorized for Color
  Use of Like Categories Yields a Uniform Display
- **High Light Output**
- **High Peak Current**
- **Excellent for Long Digit String Multiplexing**
- **Intensity and Color Selection Option**
  See Intensity and Color Selected Displays Data Sheet
- **Sunlight Viewable AlGaAs**

### Description

The 14.2 mm (0.56 inch) LED seven segment displays are designed for viewing distances up to 7 metres (23 feet). These devices use an industry standard size package and pinout. Both the numeric and ± 1 overflow devices feature a right hand decimal point. All devices are available as either common anode or common cathode.



### Devices

| AlGaAs Red HDSP-[1] | HER HDSP-[1] | Yellow HDSP- | Green HDSP- | Description | Package Drawing |
|---|---|---|---|---|---|
| H151 | 5501 | 5701 | 5601 | Common Anode Right Hand Decimal | A |
| H153 | 5503 | 5703 | 5603 | Common Cathode Right Hand Decimal | B |
| H157 | 5507 | 5707 | 5607 | Common Anode ± 1. Overflow | C |
| H158 | 5508 | 5708 | 5608 | Common Cathode ± 1. Overflow | D |
| | 5521 | 5721 | 5621 | Two Digit Common Anode Right Hand Decimal | E |
| | 5523 | 5723 | 5623 | Two Digit Common Cathode Right Hand Decimal | F |

Note:
1. These displays are recommended for high ambient light operation. Please refer to the HDSP-H10X/K12X AlGaAs and HDSP-555X HER data sheet for low current operation.

These displays are ideal for most applications. Pin for pin equivalent displays are also available in a low current design. The low current displays are ideal for portable applications. For additional information see the Low Current Seven Segment Displays data sheet.

**Part Numbering System**

5082 -X X X X-X X X X X
HDSP-X X X X-X X X X X

**Mechanical Options**[1]
00: No Mechanical Option

**Color Bin Options**[1,2]
0: No Color Bin Limitation

**Maximum Intensity Bin**[1,2]
0: No Maximum Intensity Bin Limitation

**Minimum Intensity Bin**[1,2]
0: No Minimum Intensity Bin Limitation

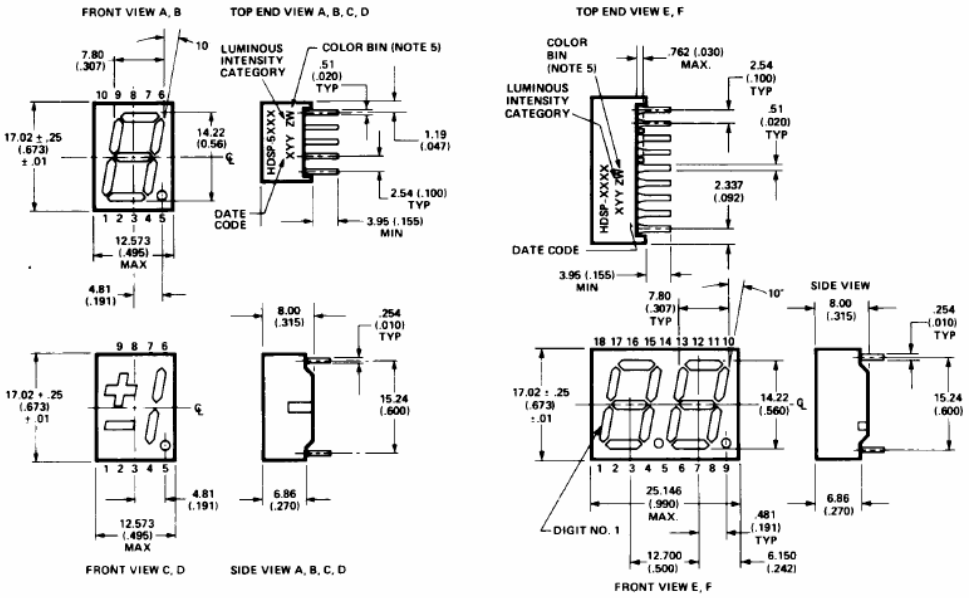
**Device Configuration/Color**[1]
1: Common Anode
3: Common Cathode

**Device Specific Configuration**[1]
Refer to Respective Datasheet

**Package**[1]
H: 14.2 mm (0.56 inch) Single Digit Seven Segment Display

Notes:
1. For codes not listed in the figure above, please refer to the respective datasheet or contact your nearest Agilent representative for details.
2. Bin options refer to shippable bins for a part number. Color and Intensity Bins are typically restricted to 1 bin per tube (exceptions may apply). Please refer to respective datasheet for specific bin limit information.

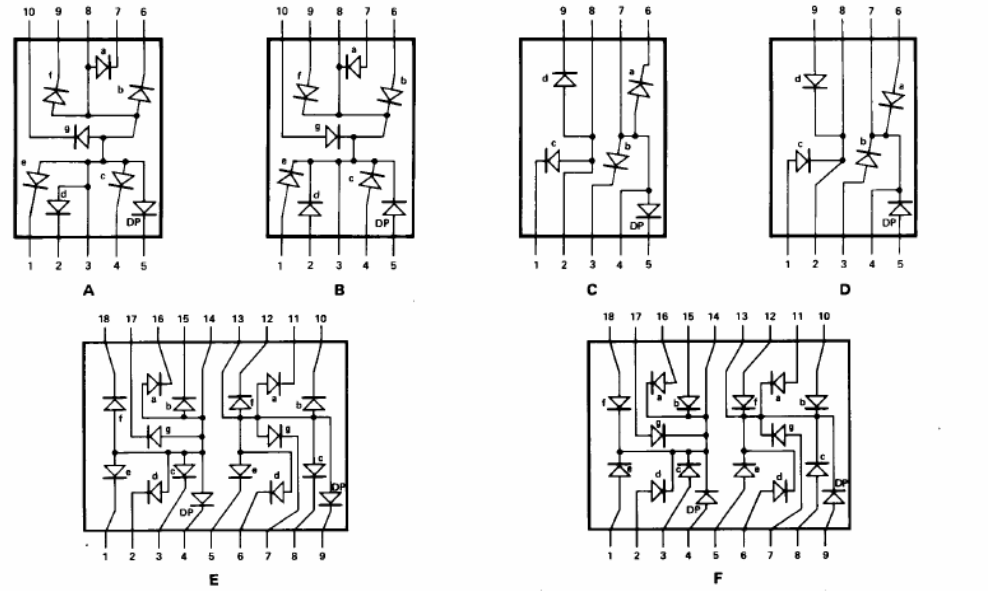
## Package Dimensions




| PIN | FUNCTION | | | | | |
|---|---|---|---|---|---|---|
| | A | B | C | D | E | F |
| 1 | CATHODE e | ANODE e | CATHODE c | ANODE c | E CATHODE NO. 1 | E ANODE NO. 1 |
| 2 | CATHODE d | ANODE d | ANODE c, d | CATHODE c, d | D CATHODE NO. 1 | D ANODE NO. 1 |
| 3 | ANODE[3] | CATHODE[4] | CATHODE b | ANODE b | C CATHODE NO. 1 | C ANODE NO. 1 |
| 4 | CATHODE c | ANODE c | ANODE a, b, DP | CATHODE a, b, DP | DP CATHODE NO. 1 | DP ANODE NO. 1 |
| 5 | CATHODE DP | ANODE DP | CATHOPDE DP | ANODE DE | E CATHODE NO. 1 | E ANODE NO. 2 |
| 6 | CATHODE b | ANODE b | CATHODE a | ANODE a | D CATHODE NO. 2 | D ANODE NO. 2 |
| 7 | CATHODE a | ANODE a | ANODE a, b, DP | CATHODE a, b, DP | G CATHODE NO. 2 | G ANODE NO. 2 |
| 8 | ANODE[3] | CATHODE[4] | ANODE c, d | CATHODE c, d | C CATHODE NO. 2 | C ANODE NO. 2 |
| 9 | CATHODE f | ANODE f | CATHODE d | ANODE d | DP CATHODE NO. 2 | DP ANODE NO. 2 |
| 10 | CATHODE g | ANODE g | NO PIN | NO PIN | B CATHODE NO. 2 | B ANODE NO. 2 |
| 11 | | | | | A CATHODE NO. 2 | A ANODE NO. 2 |
| 12 | | | | | F CATHODE NO. 2 | F ANODE NO. 2 |
| 13 | | | | | DIGIT NO. 2 ANODE | DIGIT NO. 2 CATHODE |
| 14 | | | | | DIGIT NO. 1 ANODE | DIGIT NO. 1 CATHODE |
| 15 | | | | | B CATHODE NO. 1 | B ANODE NO. 1 |
| 16 | | | | | A CATHODE NO. 1 | A ANODE NO. 1 |
| 17 | | | | | G CATHODE NO. 1 | G ANODE NO. 1 |
| 18 | | | | | F CATHODE NO. 1 | F ANODE NO. 1 |

NOTES:
1. ALL DIMENSIONS IN MILLIMETRES (INCHES).
2. ALL UNTOLERANCED DIMENSIONS ARE FOR REFERENCE ONLY.
3. REDUNDANT ANODES.
4. REDUNDANT CATHODES.
5. FOR HDSP-5600/-5700 SERIES PRODUCT ONLY.

## Internal Circuit Diagram




## Absolute Maximum Ratings

| Description | AlGaAs Red HDSP-H150 Series | HER HDSP-5500 Series | Yellow HDSP-5700 Series | Green HDSP-5600 Series | Units |
|---|---|---|---|---|---|
| Average Power per Segment or DP | 96 | 105 | 80 | 105 | mW |
| Peak Forward Current per Segment or DP | 160[1] | 90[3] | 60[5] | 90[7] | mA |
| DC Forward Current per Segment or DP | 40[2] | 30[4] | 20[6] | 30[8] | mA |
| Operating Temperature Range | -20 to +100[9] | -40 to +100 | | | °C |
| Storage Temperature Range | -55 to +100 | | | | °C |
| Reverse Voltage per Segment or DP | 3.0 | | | | V |
| Lead Solder Temperature for 3 Seconds (1.60 mm [0.063 in.] below seating plane) | 260 | | | | °C |

**Notes:**

1. See Figure 2 to establish pulsed conditions.
2. Derate above 46°C at 0.54 mA/°C.
3. See Figure 7 to establish pulsed conditions.
4. Derate above 53°C at 0.45 mA/°C.
5. See Figure 8 to establish pulsed conditions.
6. Derate above 81°C at 0.52 mA/°C.
7. See Figure 9 to establish pulsed conditions.
8. Derate above 39°C at 0.37 mA/°C.
9. For operation below -20°C, contact your local Agilent components sales office or an authorized distributor.

## Electrical/Optical Characteristics at $T_A$ = 25°C

### AlGaAs Red

| Device Series HDSP- | Parameter | Symbol | Min. | Typ. | Max. | Units | Test Conditions |
|---|---|---|---|---|---|---|---|
| H15X | Luminous Intensity/Segment[1,2,5] (Digit Average) | $I_V$ | 9.1 | 16.0 | | mcd | $I_F$ = 20 mA |
| | Forward Voltage/Segment or DP | $V_F$ | | 1.8 | | V | $I_F$ = 20 mA |
| | | | | 2.0 | 3.0 | | $I_F$ = 100 mA |
| | Peak Wavelength | $\lambda_{PEAK}$ | | 645 | | nm | |
| | Dominant Wavelength[3] | $\lambda_d$ | | 637 | | nm | |
| | Reverse Voltage/Segment or DP[4] | $V_R$ | 3.0 | 15 | | V | $I_R$ = 100 µA |
| | Temperature Coefficient of $V_F$/Segment or DP | $\Delta V_F$/°C | | -2 | | mV/°C | |
| | Thermal Resistance LED Junction-to-Pin | $R\theta_{J\text{-}Pin}$ | | 400 | | °C/W/ Seg | |

### High Efficiency Red

| Device Series HDSP- | Parameter | Symbol | Min. | Typ. | Max. | Units | Test Conditions |
|---|---|---|---|---|---|---|---|
| 55XX | Luminous Intensity/Segment[1,2,6] (Digit Average) | $I_V$ | 900 | 2800 | | µcd | $I_F$ = 10 mA |
| | | | | 3700 | | | $I_F$ = 60 mA Peak: 1 of 6 df |
| | Forward Voltage/Segment or DP | $V_F$ | | 2.1 | 2.5 | V | $I_F$ = 20 mA |
| | Peak Wavelength | $\lambda_{PEAK}$ | | 635 | | nm | |
| | Dominant Wavelength[3] | $\lambda_d$ | | 626 | | nm | |
| | Reverse Voltage/Segment or DP[4] | $V_R$ | 3.0 | 30 | | V | $I_R$ = 100 µA |
| | Temperature Coefficient of $V_F$/Segment or DP | $\Delta V_F$/°C | | -2 | | mV/°C | |
| | Thermal Resistance LED Junction-to-Pin | $R\theta_{J\text{-}Pin}$ | | 345 | | °C/W/ Seg | |

### Yellow

| Device Series HDSP- | Parameter | Symbol | Min. | Typ. | Max. | Units | Test Conditions |
|---|---|---|---|---|---|---|---|
| 57XX | Luminous Intensity/Segment[1,2] (Digit Average) | $I_V$ | 600 | 1800 | | µcd | $I_F$ = 10 mA |
| | | | | 2750 | | | $I_F$ = 60 mA Peak: 1 of 6 df |
| | Forward Voltage/Segment or DP | $V_F$ | | 2.1 | 2.5 | V | $I_F$ = 20 mA |
| | Peak Wavelength | $\lambda_{PEAK}$ | | 583 | | nm | |
| | Dominant Wavelength[3,7] | $\lambda_d$ | 581.5 | 586 | 592.5 | nm | |
| | Reverse Voltage/Segment or DP[4] | $V_R$ | 3.0 | 40 | | V | $I_R$ = 100 µA |
| | Temperature Coefficient of $V_F$/Segment or DP | $\Delta V_F$/°C | | -2 | | mV/°C | |
| | Thermal Resistance LED Junction-to-Pin | $R\theta_{J\text{-}Pin}$ | | 345 | | °C/W/ Seg | |

### High Performance Green

| Device Series HDSP- | Parameter | Symbol | Min. | Typ. | Max. | Units | Test Conditions |
|---|---|---|---|---|---|---|---|
| 56XX | Luminous Intensity/Segment[1,2] (Digit Average)` | $I_V$ | 900 | 2500 | | µcd | $I_F$ = 10 mA |
| | | | | 3100 | | | $I_F$ = 60 mA Peak: 1 of 6 df |
| | Forward Voltage/Segment or DP | $V_F$ | | 2.1 | 2.5 | V | $I_F$ = 10 mA |
| | Peak Wavelength | $\lambda_{PEAK}$ | | 566 | | nm | |
| | Dominant Wavelength[3,7] | $\lambda_d$ | | 571 | 577 | nm | |
| | Reverse Voltage/Segment or DP[4] | $V_R$ | 3.0 | 50 | | V | $I_R$ = 100 µA |
| | Temperature Coefficient of $V_F$/Segment or DP | $\Delta V_F$/°C | | -2 | | mV/°C | |
| | Thermal Resistance LED Junction-to-Pin | $R\theta_{J\text{-}Pin}$ | | 345 | | °C/W/ Seg | |

**Notes:**
1. Device case temperature is 25°C prior to the intensity measurement.
2. The digits are categorized for luminous intensity. The intensity category is designated by a letter on the side of the package.
3. The dominant wavelength, $\lambda_d$, is derived from the CIE chromaticity diagram and is that single wavelength which defines the color of the device.
4. Typical specification for reference only. Do not exceed absolute maximum ratings.
5. For low current operation, the AlGaAs HDSP-H10X series displays are recommended. They are tested at 1 mA dc/segment and are pin for pin compatible with the HDSP-H15X series.
6. For low current operation, the HER HDSP-555X series displays are recommended. They are tested at 2 mA dc/segment and are pin for pin compatible with the HDSP-550X series.
7. The Yellow (HDSP-5700) and Green (HDSP-5600) displays are categorized for dominant wavelength. The category is designated by a number adjacent to the luminous intensity category letter.